|  |
| --- |
| **校徽2**  **本科毕业论文**  （SIP-ALG解决方案）  学生姓名：冉启蕾  指导教师：赵振洋  所在学院：软件学院  所学专业：软件工程  中国·长春  2010年5月 |

**摘要**

随着IP网络的迅猛发展，传统的电信网向互联网融合的过程中，VoIP技术也越来越受到人们的关注，VoIP技术能够为服务提供商提供更多更好的服务。利用VoIP技术用户可以通过计算机网来拨打电话，VoIP技术把语音多媒体信息转换成数字信号，然后通过计算机网络进行传输，这样极大的降低了通讯成本，在VoIP技术中SIP协议比H323协议具有简单、开放、兼容和可扩展等特性，VoIP系统越来越多的采用SIP协议，SIP协议本身只是建立一个会话，而他的其他功能还是要依靠别的协议来实现，这样SIP就能够整合所有现存的协议，对于发展来说具有很大的意义，本文正是采用SIP协议进行编写的，具有很大的实用价值。

NAT/Firewall正处于互联网和私有网络的边缘，NAT/Firewall主要是用于解决现有的IP地址资源匮乏和隔绝两个网络进行一定的完全防护，传统的NAT是在IP、TCP和UDP层进行IP和端口的转化，而SIP协议是建立在TCP和UDP之上的应用层协议，在SIP协议的数据包头包含了进行路由、信令接续和建立呼叫连接必不可少的IP地址和端口信息，导致私有网络用户不能够建立正确的通话，因此SIP协议穿透NAT/Firewall成了VoIP技术急需要问题。

本文在第二章介绍了有关SIP协议的背景知识，第三章介绍了NAT/Firewall相关的背景知识，第四章提出了几种解决方案，第五章讲述了ALG的软件架构和测试的实验数据，第六章讲述了ALG的实验总结。

[**关键字**]

SIP、ALG-SIP、NAT、Firewall、RTP、SDP、SIP穿透防火墙

***abstract***

With the rapid development of IP networks, the traditional telecommunications network to the Internet, the process of integration, VoIP technology is also more and more attention, VoIP technology for service providers to provide more and better services. Users can use VoIP technology to make calls through computer networks, VoIP technology to multimedia information into digital audio signals, and then transmitted through the computer network, so that greatly reduces the communication costs, in the SIP protocol VoIP technology is simple compared with H323 protocol , open, scalable, compatible, and features, VoIP systems using SIP protocol more and more, SIP protocol itself is just a conversation, and he still have to rely on other functions to implement other protocols, so on to integrate all existing SIP The agreement, for the development of great significance for this paper is written using SIP protocol, has great practical value.

NAT / Firewall is at the edge of the Internet and private networks, NAT / Firewall is mainly used to solve the existing IP address resource shortages and the two networks will cut off the full protection of traditional NAT in IP, TCP and UDP layer the conversion of IP and port, while the SIP protocol is built on top of TCP and UDP application layer protocol header in the SIP protocol data includes the routing, signaling and the establishment of call connection connecting IP address and port essential information lead to private network users can not create the correct call, the SIP protocol through NAT / Firewall VoIP technology has become an urgent need to issue. In this paper, the second chapter on the background knowledge of SIP protocol, the third chapter of the NAT / Firewall background knowledge, the fourth chapter proposes several solutions, ALG chapter describes the software architecture and testing of experimental data , Chapter VI describes the experimental summary of ALG.

[**Key Word**]

SIP、ALG-SIP、NAT、Firewall、RTP、SDP、SIP through Firewalls

**目录**

[1、 绪论 6](#_Toc294171459)

[1.1、 课题背景 6](#_Toc294171460)

[1.2、 本文组织 6](#_Toc294171461)

[2、 SIP系统介绍 7](#_Toc294171462)

[2.1、 SIP协议简介 7](#_Toc294171463)

[2.1.1、 SIP网络元素 7](#_Toc294171464)

[2.1.2、 SIP信令定义 7](#_Toc294171465)

[2.1.3、 SIP消息格式 8](#_Toc294171466)

[2.1.4、 SIP交互流程 10](#_Toc294171467)

[2.2、 SDP协议简介 13](#_Toc294171468)

[2.3、 RTP协议简介 14](#_Toc294171469)

[3、 防火墙原理 15](#_Toc294171470)

[3.2、 NAT简介 16](#_Toc294171471)

[4、 SIP穿透防火墙方案研究 19](#_Toc294171472)

[4.1、 ALG方案 19](#_Toc294171473)

[4.2、 MIDCOM方案 19](#_Toc294171474)

[4.3、 STUN方案 20](#_Toc294171475)

[4.4、 TURN方案 21](#_Toc294171476)

[4.5、 Full Proxy方案 21](#_Toc294171477)

[4.6、 方案选取 22](#_Toc294171478)

[5、 SIP-ALG设计 23](#_Toc294171479)

[5.1、 系统流程 23](#_Toc294171480)

[5.1.1、 整个流程 23](#_Toc294171481)

[5.1.2、 Inbound Request 处理流程 24](#_Toc294171482)

[5.1.3、 Inbound Respones处理流程 24](#_Toc294171483)

[5.1.4、 OutBound Request处理流程 25](#_Toc294171484)

[5.1.5、 OutBound Response处理流程 27](#_Toc294171485)

[5.1.6、 注册请求信息 28](#_Toc294171486)

[5.2、 软件架构 30](#_Toc294171487)

[5.2.1、 Linux netfilter简介 30](#_Toc294171488)

[5.2.2、 软件架构图 30](#_Toc294171489)

[5.3、 测试数据 31](#_Toc294171490)

[5.3.1、 NAT模式 31](#_Toc294171491)

[6、 结论与展望 46](#_Toc294171492)

[6.1、 结论 46](#_Toc294171493)

[6.2、 展望 46](#_Toc294171494)

[参考文献 47](#_Toc294171495)

1. 绪论
   1. 课题背景

VoIP是将模拟信号转换成数字信号，以数据包的型式在因特网上进行实时传递。 VoIP最大的优势是能广泛地采用Internet和全球IP互连的环境，提供比传统业务更多、更好的服务。 VoIP可以在IP网络上便宜的传送语音、传真、视频、和数据等业务，如统一消息、虚拟电话、虚拟语音/传真邮箱、查号业务、Internet呼叫中心、Internet呼叫管理、电视会议、电子商务、传真存储转发和各种信息的存储转发等。

SIP是一个用来控制媒体会话建立、修改、终止的信令协议，是VoIP系统中重要的会话控制协议。媒体会话能够包含各种实时数据，例如声、影像、数据以及实时讯息。SIP是一个简单的基于纯文本的信令协议，而实际数据传输是通过RTP协议来实现的。

NAT/防火墙为私网提供统一的对外出口，从而隐藏内部网络的拓扑结构，提高了私网的安全性。但这也给私网的远程控制应用带来很大的麻烦。对于NAT其功能是在公网IP地址及端口和私网IP地址及端口间进行映射，工作在传输层，它只对TCP/UDP包头中的地址、端口进行修改，而SIP协议需要在信令消息中内嵌IP地址和端口号，这些地址、端口在应用层上才可见，因此NAT不会对其中的地址信息进行修改，导致信令消息中的IP地址和端口不能指向正确的地址，因而通信也不能正常进行;对于防火墙对公网打开的端口通常是固定（防火墙不会在运行过程中动态的打开或者关闭这些端口)，且数目有限。而基于SIP构建的私网的远程控制应用要求防火墙不但能够提供对信令协议的代理功能，而且要求防火墙能够在通信过程中动态的打开一些端口进行媒体流数据的交流，现有的防火墙难以满足这个要求。

* 1. 本文组织

本文是研究的是基于Linux的SIP协议的应用层网管，第一章是绪论，第二章介绍了基于SIP协议的VoIP系统的各个协议，第三章介绍了防火墙和NAT的基本原理，第四章介绍SIP协议穿透防火墙的几种实现方案的研究，第五章介绍了SIP-ALG实现方案的软件流程和软件架构，并在此基础上进行的实验数据分析，第六章是该解决方案的实验总结。

定义缩写

ALG­ - Application Level Gate（应用层网关）

NAT – Net Address Translate（网络地址转换）

Call Leg（呼叫腿），它由SIP协议中Call-ID，To tag，From tag头域组成的单元组

SDP – Session Description Protocol（会话描述协议）

SIP – Session Initiation Protocol（会话发起协议）

UAC - User Agent Client（用户代理客户端）

UAS – User Agent Server（用户代理服务器）

UA – User Agent（一个逻辑实体的概念，包含UAC和UAS）

Proxy Server（代理服务器）

1. SIP系统介绍
   1. SIP协议简介

SIP（SessionIn itiation Protocol）协议是一个用于建立．更改和终止多媒体会话的应用层控制协议，它是由Internet工程任务组（IETF）提出的IP电话信令协议。正如其名字所隐含的SIP信令用于发起会话，它能控制多个参与者参加的多媒体会话的建立和终结，并能动态调整和修改会话属性，如会话带宽要求、传输的媒体类型（语音、视频和数据等）、媒体的编解码格式、对组播和单播的支持等。

* + 1. SIP网络元素

SIP系统是基于服务器-客户端的模式建立，其中有用户代理（User Agent Client UAC）和用户代理服务器（User Agent Server UAS），用户代理是会话的发起者或是接受者，用户代理可以是真实的SIP话机，也可以是软件模拟的软终端。

用户代理服务器包含很多种服务器，有注册服务器、代理服务器和重定向服务器等它们分别完成不同的功能，也可以几种类型的服务器合成一个服务器以完成不同的功能。

注册服务器是用于完成用户代理注册，注册服务器会记录下用户的地址信息。

代理服务器是用于对SIP信令进行转发和路由，也可模拟用户代理发起SIP请求或响应。

重定向服务器不会对SIP信令进行转发而是把请求地址映射成几个新的地址，使得呼叫着可以直接联系下一服务器。

* + 1. SIP信令定义

SIP消息有两种请求消息和响应消息，请求消息是客户机或服务器发起请求的消息，响应消息时响应请求消息的消息。

1. SIP请求消息

SIP 请求消息的起始行由方法、Request-URI、SIP协议版本号三部分顺序组成，三者之间用空格隔开。SIP 共规定了六种方法：INVITE，ACK，CANCEL，OPTIONS，BYE，REGISTER。其中 INVITE 和 ACK 用于建立呼叫，完成三次握手，或者用于呼叫建立以后改变会话属性，其中 ACK 是在接收到 INVITE 的最终响应之后发出的确认，它只与 INVIT捆绑使用；BYE 用于结束一次会话；OPTIONS 用于查询用户代理服务器的能力；CANCEL用于取消已经发出但尚未完成的请求；REGISTER 用于客户机向注册服务器注册用户位置等信息。

1. SIP响应消息

SIP 响应消息的起始行由 SIP 协议版本号、状态码和用自然语言描述的原因组成。状态码由3个阿拉伯数字组成，用于计算机判断具体响应的结果，原因部分则是出于帮助 SIP 用户更好的理解响应消息的目的而设计的。SIP 中定义了6类状态码，由第一位数字来区分，可以方便的进行扩展。6类标示响应类型的状态码分别是：

（1）1xx：提示信息，表示请求已收到，正在处理。

（2）2xx：请求被成功接收并处理。

（3）3xx：重定向，为完成请求功能需进一步执行动作。

（4）4xx：客户端错误，如请求消息语法错误、服务器不能完成操作等。

（5）5xx：服务器端错误，服务器不能处理明显的合法请求。

（6）6xx：全局错误，请求不能在任何服务器中处理。

其中常见的信息状态码有 100 Trying，180 Ringing，181 Call Is Being Forwarded

等，表示呼叫成功的状态码是 200 OK。

响应消息状态码

表2 - 1

|  |  |  |  |
| --- | --- | --- | --- |
| 临时响应1xx | 100 Trying | 180 Ringing | 181 Call Is Beging |
| 成功响应2xx | 200 OK |  |  |
| 重定向3xx | 300 Multiple Choice | 301 Moved Permanenty | 302 Moved Temporarily |
| 客户端错4xx | 400 Bad Request | 401 Unauthorized | 402 Payment Required |
| 403 Forbidden | 404 Not Found | 405 Method Not Allowed |
| 406 Not Acceptable | 407 Proxy Aut-enticAti  -on Required | 408 Request Timeout |
| 410 Gone | 413 Request Entity Too  Large | 414 Request-URI  Too Large |
| 415 Unsupported  Media Type | 416 Unsupported URI  Scheme | 420 Bad Extension |
| 421 Extension  Required | 423 Interval Too Brief | 480 Temporarily  not available |
| 481 Call  Leg/Transaction Does Not Exist | 482 Loop Detected | 483 Too Many Hops |
| 484 Address  Incomplete | 485 Ambiguous | 486 Busy Here |
| 487 Request  Terminated | 488 Not Acceptable  Here | 491 Request Pending |
| 493 Undecipherable |  |  |
| 服务端错5xx | 500 Interval Server  Error | 501 Not Implemented | 502 Bad Gateway |
| 503 Service  Unavailable | 504 Server Time-out | 505 SIP Version not  Supported |
| 513 Message Too  Large |  |  |
| 全局错误6xx | 600 Busy  Everywhere | 603 Decline | 604 Does not exist  Anywhere |
| 606 Not Acceptable |  |  |

* + 1. SIP消息格式

SIP消息分为从客户端到服务器端的请求消息和从服务器端到客户端的响应消息。两类消息都遵从通用的消息格式，由起始行、一个或多个头域、一个空行和可选的消息体顺序构成。SIP消息格式如下：

表2 - 2

|  |  |
| --- | --- |
| Start - Line | 起始行 |
| \*Message - Header | 一个或多个头域 |
| CRLF | 空行 |
| [ Message – Body ] | 消息体 |

起始行：每个消息都由起始行开始，起始行可以使请求行和状态行（响应行），请求行是由请求方法类型加协议版本号组成，在状态行是由版本号加状态码组成。

SIP头：是由一个或多个头域组成，头域是按照“<头域名>:<值>”的方式组成，且一行一个头域，SIP主要支持的头域将会下面做详细介绍。

消息体：是用于描述会话初始的信息，如媒体交互的IP地址和端口、带宽容量和采样率等信息，消息体可以在有也可以没有，可以在请求消息中也可以在响应消息中。

SIP头域比较多想详细了解请参考RFC文档，这里只对几个重要的头域做介绍。SIP头域可以分为四类头域general-header、entity-header、request-header和response-header

1. general-header类

为描述消息基本属性的通用头域，可用于请求消息或响应消息；通用头域的域名只有在协议版本改变时才可有效地扩展。不过，通信中的所有方均认为是“通用头域”的新的头域也可认为是通用头域。不被认可的头域作为实体头域。

Call-ID: 通用头域唯一标识一个特定的请求或者一个特定客户的所有登记。

Form：请求和响应必须包含From通用头域，指示请求的初始者。

To：通用头域说明了请求的接收者。

Via：Via头域指示请求迄今为止所走的路径。它防止了请求的循环，同时确保了响应（回答）沿同样的路径返回，这一点可以通过防火墙遍历和其他的异常路径情况提供帮助。

Contact：通用头域可出现在INVITE、ACK和REGISTER请求中，1xx、2xx、3xx和485响应中。通常，它提供了一个URL，用户可以通过此URL来进行进一步的通信。

Cseq：对于每一个请求，客户必须使用Cseq（Command sequence）通用头域。此头域包含了请求方式和一个提出请求的客户所选定的十进制序列数，在同一个Call-ID中此Cseq值唯一。此序列数必须为一个32位的无符号整数，它的初始值是任意的，但必须小于等于2\*\*31。

Expires：Expires头域给出了消息内容活动的日期和时间。此头域只用于INVITE、REGISTER方式。

1. entity-header类

用于描述消息体内容的长度、格式和编码类型等属性，可用于请求消息或响应消息。实体头域定义了消息体信息之后的内容（如：Content-Length、Content-Type、Content-Encoding），或者如果没有消息体，则定义请求所指示的资源。

Content-Encoding：Content-Encoding=(“Content-Encoding” | “e”)”:” 1#content-coding Content-Encoding实体头域作为“media-type”的一个修饰语。它的值指示适用于实体消息体的其他的内容编码，指示为了获得Content-Type头域所给出的media-type，必须使用的编码方案。Content-Encoding主要用于压缩消息体，而不丢失它底层的媒体类型的标识。

Content-Length：实体头域指示消息体的长度。形式上以八个比特为一个字节。Content-Length = (“Content-Length” | “l”)”:” 1\*DIGIT

应用程序应该使用此域来指示所传送的消息体的大小，而不管实体所用的媒体类Content-Length的值应为非负数，0表示没有消息体。

Content-Type：实体头域指示发送给接收者的消息体的媒体类型。

Content-Type=（“Content-Type”| “c”）“：”media-type

1. request-header类

为请求头域，只可用于请求消息，它用来传递有关请求或客户机本身的一些附加信息，对请求进行补充说明。客户将关于请求和关于客户自己的其他信息传送给服务器。这些域类似于请求的变量，语义上相当于可编程语言方式调用的参数。请求头域的扩展与通用头域相同。

Subject：请求头域提供了一个摘要，或者指示了呼叫的实际情况，使得不必分析通话描述便可过滤呼叫。（当然，通话描述不必使用与邀请同样的标题）

Subject = （"subject" | "s"）"："\*TEXT-UTF8

User-Agent：通用头域包含了关于发送初始请求的客户用户代理的消息。此头域用于统计目的，跟踪违反协议的情况、用户代理的自动认可的情况，以便在编制响应时避免特定用户代理的限制。用户代理应在请求中包含此头域。

User-Agent = "User-Agent" ":" 1\*( product | comment )

Organization：通用头域表明了发送请求或者响应的实体所属的组织。它可以由位于某组织边界的代理来加入。客户软件可以使用此头域来过滤呼叫。

Organization ="Organization" "："\*TEXT-UTF8

Contact：通用头域可出现在INVITE、ACK和REGISTER请求中，1XX、2XX、3XX和485响应中。通常，它提供了一个URL，用户可以通过此URL来进行进一步的通信。INVITE和ACK请求：Contact域表明请求从哪个位置发起；这允许主叫方直接向被叫方发送未来的请求，如BYE，而不是通过一系列的代理。由于所想要的地址可能是代理的地址，所以只Via头域并不够。

Authorization：客户通过一个Authorization头来重新测试请求。

1. response-header类

为响应头域，只可用于响应消息，它用来传递有关响应的附加信息，对响应进行补充说明，如有关服务器的信息和需要作出的下一步动作的提示等；允许服务器发送关于响应的无法放在Status-Line中的其他信息。这些头域给出了关于服务器和关于进一步访问由Request-URL指示的资源的信息。响应头域的扩展与通用头域相同。

Proxy-Authorization：请求头域允许客户向要求验证的代理来鉴别自己。

Retry-After：头域用在503（Service Unavailable）响应中，向提出申请的客户指示，此服务预计多长时间无效。

Server：响应头域包含了关于UAS用来处理请求的软件的信息。如果响应通过代理来前转，那么代理禁止修改此Server响应头域，它应该包含一个Via头域。

Warning：响应头域中包含了关于响应状态的其他信息。

* + 1. SIP交互流程

SIP交互流程有中终端注册、终端注销、连接建立、会话更改流程、连接释放、呼叫保持、呼叫前传和呼叫转移等众多业务，在此只介绍终端注册、终端注销、连接建立、会话更改流程和连接释放这几种基础的业务流程，其他更复杂的业务流程是建立在基础业务流之上的，有兴趣的读者可以参考RFC文档。

1. 终端注册流程

流程说明：

1. 用户第一次使用终端（Phone1）的时候，Phone1向Proxy发送注册请求用于注册；



图2 - 1终端注册流程

1. Proxy对Phone1进行认证，如果Phone1不在数据库中便回送包含认证令牌的401 Unauthorized响应给Phone1；
2. Phone1将用户输入的用户名和密码经过认证令牌加密后，再次将注册请求发送给Proxy；
3. Proxy将注册消息中的用户信息解密，验证其合法后，将该用户信息登记到数据库中，并向Phone1返回200 OK的成功响应消息。
4. 终端注销流程



图2 - 2终端注销流程

流程说明：

1. Phone1向Proxy发送Register消息注销，其中超市字段（expire）字段置为零。
2. Proxy收到200 OK响应后，并将用户注销。
3. 连接建立流程

流程说明：

1. Phone1向Proxy发起Invite请求；
2. Proxy根据Phone2注册时留下的IP地址和端口信息将 Invite 消息的转发给Phone2；
3. Proxy向Phone1发送100 Trying，告诉Phone1请求正在被处理中；
4. Phone2向Proxy发送 100 Trying，告诉Phone1请求正在被处理中;
5. Phone2振铃，并向Proxy发送 180 Ringing振铃信息；
6. Proxy向Phone1转发180 Ringing信息；



图2 - 3连接建立流程

1. 被叫用户摘机，Phone2向Proxy发送200 OK表示invite请求成功；
2. Proxy向Phone1转发200 OK表示invite请求成功；
3. Phone1收到200 OK后向Proxy发 ACK进行确认；
4. Proxy将ACK消息转发给Phone2表示确认；
5. Phone1和Phone2之间建立通信连接，开始通话；
6. 会话更改流程



图2 - 4会话更改流程

1. 呼叫释放流程

呼叫释放流程有正常呼叫释放流程、网络异常呼叫释放流程。

正常的呼叫释放流程：



图2 - 5呼叫释放流程

1. 通话结束后，被叫用户挂机，Phone2向Proxy发送 Bye 消息；
2. Proxy转发 Bye 消息至Phone1；
3. 主叫用户挂机后，Phone1向Proxy发送确认挂断响应消息 200 OK；
4. Proxy转发200 OK给Phone2。

网络异常呼叫流程

网络异常呼叫流程有被叫忙情况下的呼叫释放、被叫无应答情况下的释放流程（主叫放弃）和被叫无应答情况下的释放流程（被叫不应答）。在此只介绍被叫忙情况下的呼叫释放流程，有兴趣的读者可以参考RFC文档。

被叫忙情况下的呼叫释放流程



图2 - 6被叫忙情况下的呼叫释放流程

1. 用户A发起一路呼叫，Phone1向Proxy发送 Invite 请求。
2. Proxy收到invite请求后向Phone2转发该 Invite 消息。
3. Proxy向Phone1回送 100 Trying，告知Phone1呼叫正在处理。
4. Phone2收到invite请求后，被叫忙，Phone2向Proxy送 486 被叫忙响应。
5. Proxy向Phone1转发486消息。
6. Phone1向Proxy回送 ACK 确认消息。
7. Proxy向Phone2送 ACK 确认消息。
   1. SDP协议简介

SDP也是MMUSIC工作组的一个产品,在MBONE内容中用得很多。其目的就是在媒体会话中，传递媒体流信息，允许会话描述的接收者去参与会话。

SDP协议包含以下方面的内容：

1. 描述会话目的和会话名称
2. 描述会话保持的时间
3. 描述会话中进行交互的媒体信息，包括：

媒体类型(视屏、音频、数据、传真等)

传输协议(RTP/UDP/IP、H.320)

媒体格式(H.261 video、MPEG video)

多播或远端（单播）地址和端口

1. 描述为接收媒体信息而需要的信息，如IP地址、端口和数据格式等等
2. 描述可以使用的带宽信息
3. 描述可以信任的接洽信息（Contact information）
   1. RTP协议简介

实时传送协议（Real-time Transport Protocol或简写RTP，也可以写成RTTP）是一个网络传输协议，它是由IETF的多媒体传输工作小组1996年在RFC 1889中公布的。

RTP协议详细说明了在互联网上传递音频和视频的标准数据包格式。它一开始被设计为一个多播协议，但后来被用在很多单播应用中。RTP协议常用于流媒体系统（配合RTCP协议），视频会议和一键通（Push to Talk）系统（配合H.323或SIP），使它成为IP电话产业的技术基础。RTP协议和RTP控制协议RTCP一起使用，而且它是建立在用户数据报协议上的。

RTP 本身并没有提供按时发送机制或其它服务质量（QoS）保证，它依赖于低层服务去实现这一过程。 RTP 并不保证传送或防止无序传送，也不确定底层网络的可靠性。 RTP 实行有序传送， RTP 中的序列号允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如：在视频解码中，就不需要顺序解码。

RTP 由两个紧密链接部分组成：

RTP ― 传送具有实时属性的数据；

RTCP ― 监控服务质量并传送正在进行的会话参与者的相关信息。RTCP 第二方面的功能对于“松散受控”会话是足够的，也就是说，在没有明确的成员控制和组织的情况下，它并不非得用来支持一个应用程序的所有控制通信请求。

1. 防火墙原理
   1. 防火墙简介
      1. 防火墙定义

防火墙是一种控制不同网络之间相互访问的设备，它能够监测不同网络之间进行访问的数据，对被保护的网络构成安全威胁的数据进行限制，并对有些数据流进行更改，尽可能的控制外部网络对内部网络的访问，使内部受保护网络的信息、结构和运行状态不被外部探知，从而架起一道安全屏障保护内部网络。如图3-1为防火墙的部署环境，私有网络和因特网之间通过防火墙进行互访，在防火墙上能对外部网络和内部网络之间互访的数据进行流量控制和数据监控，从而隔绝从因特网上对私有网络的非法访问，很大程度的屏蔽掉潜在的安全风险。



图3 - 1

* + 1. 防火墙分类：

防火墙有两种，硬件防火墙和软件防火墙，他们都能起到保护作用并筛选出网络上的攻击者。

软件防火墙工作于系统接口与NDIS之间，用于检查过滤由NDIS发送过来的数据，在无需改动硬件的前提下便能实现一定强度的安全保障，但是由于软件防火墙自身属于运行于系统上的程序，不可避免的需要占用一部分CPU资源维持工作，而且由于数据判断处理需要一定的时间，在一些数据流量大的网络里，软件防火墙会使整个系统工作效率和数据吞吐速度下降，甚至有些软件防火墙会存在漏洞，导致有害数据可以绕过它的防御体系，给数据安全带来损失，因此，许多企业并不会考虑用软件防火墙方案作为公司网络的防御措施，而是使用看得见摸得着的硬件防火墙。

硬件防火墙是一种以物理形式存在的专用设备，通常架设于两个网络的驳接处，直接从网络设备上检查过滤有害的数据报文，位于防火墙设备后端的网络或者服务器接收到的是经过防火墙处理的相对安全的数据，不必另外分出CPU资源去进行基于软件架构的NDIS数据检测，可以大大提高工作效率。

* + 1. 防火墙技术

传统意义上的防火墙技术分为三大类，包过滤、应用代理和状态监视，无论一个防火墙的实现过程多么复杂，归根结底都是在这三种技术的基础上进行功能扩展的。

1. 包过滤技术

包过滤技术是一种简单、有效的安全控制技术，它通过在网络间相互连接的设备上加载允许、禁止来自某些特定的源地址、目的地址、TCP端口号等规则，对通过设备的数据包进行检查，限制数据包进出内部网络。包过滤的最大优点是对用户透明，传输性能高。但由于安全控制层次在网络层、传输层，安全控制的力度也只限于源地址、目的地址和端口号，因而只能进行较为初步的安全控制，对于恶意的拥塞攻击、内存覆盖攻击或病毒等高层次的攻击手段，则无能为力。

包过滤防火墙一般在路由器上实现，用以过滤用户定义的内容，如IP地址。包过滤防火墙的工作原理是：系统在网络层检查数据包，与应用层无关。这样系统就具有很好的传输性能，可扩展能力强。但是，包过滤防火墙的安全性有一定的缺陷，因为系统对应用层信息无感知，也就是说，防火墙不理解通信的内容，所以可能被黑客所攻破。

1. 应用代理技术

应用网关防火墙检查所有应用层的信息包，并将检查的内容信息放入决策过程，从而提高网络的安全性。然而，应用网关防火墙是通过打破客户机／服务器模式实现的。每个客户机／服务器通信需要两个连接：一个是从客户端到防火墙，另一个是从防火墙到服务器。另外，每个代理需要一个不同的应用进程，或一个后台运行的服务程序，对每个新的应用必须添加针对此应用的服务程序，否则不能使用该服务。所以，应用网关防火墙具有可伸缩性差的缺点。

1. 状态监视技术

状态检测是比包过滤更为有效的安全控制方法。对新建的应用连接，状态检测检查预先设置的安全规则，允许符合规则的连接通过，并在内存中记录下该连接的相关信息，生成状态表。对该连接的后续数据包，只要符合状态表，就可以通过。这种方式的好处在于：由于不需要对每个数据包进行规则检查，而是一个连接的后续数据包（通常是大量的数据包）通过散列算法，直接进行状态检查，从而使得性能得到了较大提高；而且，由于状态表是动态的，因而可以有选择地、动态地开通1024号以上的端口，使得安全性得到进一步地提高。

状态检测防火墙基本保持了简单包过滤防火墙的优点，性能比较好，同时对应用是透明的，在此基础上，对于安全性有了大幅提升。这种防火墙摒弃了简单包过滤防火墙仅仅考察进出网络的数据包，不关心数据包状态的缺点，在防火墙的核心部分建立状态连接表，维护了连接，将进出网络的数据当成一个个的事件来处理。可以这样说，状态检测包过滤防火墙规范了网络层和传输层行为，而应用代理型防火墙则是规范了特定的应用协议上的行为。

* 1. NAT简介

NAT的发展是为了解决两个问题。首先，在因特网上可以用的公网IP地址日益接近耗尽；其次，因为网络间的相互连接，有可能管理员在某个网络分配的一组地址，而其他人也会使用，最典型的环境是当两个公司各自的网络需要合并的时候。NAT可以通过一种机制把任意数量的IP地址转换成另一个范围的IP地址，或者在某些情况下转换成一个单一的或者更小范围的IP地址，以此来解决上述两个问题。

为了解决可用地址日益枯竭的问题，NAT可以用来转换成百上千的IP地址到几个甚至是单一的地址，这样一来，公司就能够实现让其所有主机都能访问因特网，而不需要分配成成百上千的因特网地址。为解决无效网络或者很多情况下的重复网络问题，NAT可以实现让每个网络看上去都是一个完全不同的网络，图说明了NAT的工作进程和因特网的连接性。



图3 – 2 NAT转换实例

在这个例子中，当主机A试图去访问因特网时，防火墙将请求报文中的源地址192.168.88.100转换成另一个源地址200.200.155.167，然后将数据包传递到因特网中。然后，防火墙会在其转换表中存下这个转换条目，以便处理回程流量。当主机B收到该数据，它认为是在和200.200.155.167通信，所有发出回应流量。当防火墙收到这个回程流量后，参考其转换表，确定该流量应该转发给192.168.88.100.于是防火墙重新打包这个报文，这是将目的地址改为192.168.88.100，然后传递出去。这样一来，主机A和主机B就可以相互通信了，但是他们并不知道其中有NAT的参与。

因为NAT有效的隐藏了所使用的真实地址，很多网络将其和私有IP地址结合使用。私有IP地址在RFC 1981中被定义，他们是一组事先定义好的不能在因特网中使用的IP地址，因为这些地址不能够在因特网中被路由。因为NAT阻止了基于因特网连接的主机去探知在NAT路由后面所使用的IP地址，所以组织和单位通常使用私有IP地址去实施网络，以便可以不需要关心因特网或者其他来自网络带来的影响。RFC 1918定义的IP地址如下：

* 10.0.0.0/8
* 172.16.0.0/12
* 192.168.0.0/16
  + 1. NAT实施

有四种主要的NAT实施方法。它们实现相同的功能，即把流量的地址从一个IP转换为另一个IP地址，不过它们的转换方式不尽相同。这四种方法如下

1. 静态NAT——静态NAT有时候也被称为传统的NAT，它将一个IP地址映射到另一个IP地址。因此静态NAT要求转换和被转换的地址数量相同。基于这个原因，静态NAT对于节省IP地址数量去访问一个网络或者因特网来说，并不是一种高效的方法。
2. 动态NAT——动态NAT的功能和静态NAT类似，但是他不需要每一个IP地址都要一对一的转换，有一个动态IP地址池被用来实施转换。这样一来就可以减少所使用的IP地址数量，因为地址池的大小可以比需要被转换的IP地址数量小。
3. 端口地址转换——无论是静态转换还是动态NAT，都是将一个地址转换成另一个地址，而端口地址转化（PAT）允许将一组IP地址转换为一个单一IP地址。这是通过使用TCP或者UDP端口来实现的。执行转换的路由器或者防火墙会建立一张NAT表，但是与分配IP地址不同的是，它分配一个端口号给出站通信流量。当收到某个端口号得回应后，该路由器或者防火墙逆向执行上述过程。
4. 双向NAT——在很多情况下，NAT用来单向地址转换数据，通常是从内部或者被保护的网络到外部或者未被保护的网络转换。双向NAT可以不管流量的方向而实施NAT。
5. SIP穿透防火墙方案研究
   1. ALG方案

ALG是应用级网关(Application Level Gateway)的简称。对十NAT而言对报文的地址转换只在网路层和传输层进行，而对报文载荷中的地址信息则是无能为力的，对于SIP、H323等应用层协议，在后续的业务中需要用到报文载荷中的IP地址和端口信息进行数据传输，由于没有对载荷中的数据做相应的转换，而私网IP地址不能够在公网上路由，这样就会导致业务不能够正常进行。正是因为NAT对载荷中的信息不能够进行转化，一方面会对应用层得业务造成影响，另一方面还会是私网内部主机暴露在公网上。

解决这些特殊协议的转换问题的一个方法就是在NAT实现中采用应用级网关（ALG），应用级网关是作为NAT的一个功能增强模块出现的，在网络层和传输层识别出这些特殊应用的协议，然后转交给ALG进行处理，ALG把载荷数据中的IP地址和端口进行转换，并建立相应的映射关系，从而达到穿透NAT的目的。

ALG的处理方案只需要对NAT网关进行升级即可，而且针对不同的多媒体业务，它的实现主体思想都是相同的，不同的只是要详细了解所要处理协议报文的格式，清晰理解报文载荷中地址的位置，再做以相应的处理。

* 1. MIDCOM方案

MIDCOM 的框架是采用可信的第三方（MIDCOM Agent）对 Middlebox（NAT/ALG）进行控制的机制，应用业务识别的智能也由Middlebox 转移到外部的 MIDCOM Agent 上，因此应用协议对 Middlebox 是透明的。

下面以SIP协议为例讲述下MIDCON的工作流程，这儿主要是讲解MIDCOM架构的工作流程，对与SIP Proxy的交互过程在此不做介绍，详细介绍请参考本文的第五章，如图4 – 1。



图4 - 1

1. Phone1向MIDCOM Agent发起Invite请求，MIDCOM Agent就和Middlebox进行协商应用层NAT服务需要的IP地址和端口，并返回给MIDCOM Agent。
2. MIDCOM Agent收到Middlebox返回的NAT服务信息后，把Invite请求中的相关的IP地址和端口改成Middlebox返回的IP地址和端口。然后MIDCOM Agent 把Invite发给NAT/Firewall。
3. NAT/Firewall收到Invite只在网络层和传输层做NAT转换后就可以直接发给Phone2。
4. Phone2收到Invite请求后恢复一个200 OK给NAT/Firewall，NAT/Firewall根据NAT转换规则直接把数据包发给Phone1。
5. Phone1收到200 OK后恢复一个ACK的响应给MIDCOM Agent，MIDCON Agent又会重复（1）—（3）中的过程。
6. Phone2收到ACK后，连接就建立起来了。

MIDCOM架构通过将应用程序智能由中间盒上转移到相对独立的第三方MIDCOM Agent上, 从而简化中间盒的设计和维护。但是由于引入了第三方, 增加了额外的通信, 当系统很少时, 比如仅仅一两百个终端这样的系统, 这种方法并不能带来什么好处。从安全性考虑 MIDCOM 方式可以支持控制报文的加密，可支持媒体流的加密，因此安全性比较高。

* 1. STUN方案

STUN协议用来帮助位于NAT之后的实体发现NAT 的存在, 判断NAT 的类型, 从而得到由NAT分配的绑定映射地址。STUN无需对NAT进行改变, 适用于处理在应用程序实体与公网间有多重NAT 的情况 。

STUN由两部分组成STUN Client和STUN Server，STUN Client部署位于内网一般集成在User Agent Client中，STUN Server部署于外网，如图4 – 2。



图4 - 2

STUN Client向STUN Server发起UDP的数据包，STUN Server收到STUN Client发来的数据包后，会向STUN Client回复一个UDP数据包（其中包含收到UDP数据包的源地址），STUN Client收到回复包后就知道了NAT外网的地址，然后将该地址填入以后呼叫协议的UDP负载中，并告诉对端本端的RTP连接地址和端口号，由于STUN协议在NAT上为RTP接受地址做好了NAT映射，因此流媒体就可以穿透NAT了。

STUN的优点是不需要改动现有的NAT/Firewall架构，并能很好的适应多级NAT架构，但是也有很多局限性

1. 需要NGN客户端集成STUN Client的功能。
2. 只能对UDP进行穿透，不能实现对TCP的穿透，如H323。
3. 无法穿透对称NAT类型的防火墙。
   1. TURN方案

TRUN 全称Traversal Using Relay NAT即通过Relay方式穿透NAT，TRUN跟STUN解决问题的方式比较相似，也是事先通过某种方式获得私网对应的公网IP（TURN获得的公网IP为TURN Server上的地址），将负载信息中描述地址的信息填充为公网的IP地址和端口，以实现NAT穿透。

TURN是通过 Relay 方式穿透NAT，TRUN的工作方式是通过TRUN Server给私网的TRUN Client分配TRUN Server的IP地址和端口作为接受地址，即私网的数据包要经过TRUN Server进行relay转发发，这种应用模式不仅具有了STUN的优点，而且还能穿透对称NAT，同时还支持TCP穿透（支持H323协议），还能控制RTP的端口控制。

这种应用方式的局限在于需要NGN客户端支持TRUN Client功能。

* 1. Full Proxy方案

Full Proxy方案是指通过对私网内用户呼叫的信令和媒体同时做Relay来实现出口NAT/Firewall的穿越，信令Proxy和媒体Proxy可以在同一台机器上实现也可以分开实现，一起实现叫Full Proxy，Full Proxy一般部署在网路的边缘或者汇聚点上，如图4 – 3。



图4 - 3

信令Proxy：Proxy设备对于用户而言是软交换系统，即用户的注册信息和呼叫信息都经过Proxy，然后转发给软交换系统，对于软交换系统Proxy就是客户端，软交换系统把注册信息和呼叫请求的响应信息都发给Proxy，Proxy负责转发给真实的用户。

媒体Proxy：Proxy设备是私网和公网之间媒体信息的必经之处，所有内网用户和外界用户进行通信的媒体数据都会经过Proxy设备，Proxy设备在此对数据包进行检验，将合法的数据包选择不同的转发策略进行转发，比如NAT转换策略。

在Full Proxy方式下，无需现有NAT做任何改动，可采用普通的设备，同时私网内的终端无需支持STUN和TURN协议，即可开展NGN业务，这是它较大的一个优势。其局限性是同TURN一样，增加了包的延时和丢包的可能性。

* 1. 方案选取

在比较了以上几种多媒体穿透NAT的解决方案之后，ALG的方案很明显没有其他方案针对性墙，性能不高而且还需要升级当前的NAT设备等缺陷，但是ALG方案实现简单且性价比高，也有比较高的安全性，并且现在大部分NAT设备都是基于软件的NAT转换，所以在此选择了ALG方案来实现SIP穿透NAT/Firewall。

1. SIP-ALG设计
   1. 系统流程
      1. 整个流程

ALG收到一个SIP消息会判断是哪种类型的消息，根据消息的类型和方向进行处理。当接收或发送请求时ALG会记录SIP消息这些字段(只记录从内网发出的),后面就简称为请求信息记录RMR,可以将这些消息放到连接跟踪里。执行到SIP ALG是NAT已经做完了，只需要根据NAT结果修改SIP包。目前的设计是基于内核ALG框架。

表5 - 1

|  |
| --- |
| Register的Cseq |
| Invite的Cseq |
| Call-ID |

Via和Contact头域可能不携带端口号，使用默认506

处理流程：



图5 - 1整个流程

* + 1. Inbound Request 处理流程

一个UAC在注册的时候在设备上留下NAT信息，Inbound Request消息可以通过这个NAT信息进入，找到发出注册的客户端，实现外网用户呼叫内网用户。



图5 - 2 Inbound Request 处理流程

* + 1. Inbound Respones处理流程

处理过程：获取响应消息中的Cseq头域确定是对哪个方法的响应。

1. Register请求，见注册过程。
2. Invite请求，需要匹配前面发出请求时的Cseq，根据响应消息码确定session是否建立成功。如果为200响应表示建立成功，如果为100-199临时响应，此时不能确定是否建立成功，其他响应表示session建立失败，直接删除发出请求时建立的媒体信息期待。
3. Update或Prack请求，如果为100-199的响应表示建立成功，其他表示失败，直接删除建立的媒体信息期待。
4. 根据配置对不支持的方法决定是否要丢掉。



图5 - 3 Inbound Respones处理流程

* + 1. OutBound Request处理流程

1. 对不支持的请求方法，根据配置将其直接丢掉或者不做特殊处理直接发出去。
2. 如果是Register注册请求，参见注册过程。
3. 如果为Invit请求，记录Cseq头域，用来匹配最终的响应消息的Cseq。
4. Bye和Cancel方法，结束或取消建立Session，既然都决定结束或取消会话，那就直接删除之前建立的媒体信息期待。
5. 其他方法，对于像Ack，Prack，Update方法都可能携带SDP消息体，所以需要为这些SDP的媒体信息建立期待。保证对端发送的RTP，RTCP信息可以通过。



图5 - 4 OutBound Request处理流程

SIP规范上说via头域功能是接受响应消息，对于UAC发出请求使用一个端口，而使另外一个端口去接受请求。通过抓包发现都是使用发送请求的那个端口，作为回复端口。但是为了满足协议的要求，需要能够处理这种情况，需要via头域建立一个期待结构，以便收到响应消息时，能够处理。

SIP规范上说contact头域功能（非register请求，contact对于register请求特殊含义，参见注册流程）对端的UAS可以直接将后续的请求发送到这个地址（对于内网用户contact大多与via的头域端口号是一样的）可以不再通过代理服务器。需要为这种情况建立期待，保证后续的对端请求可以直接通过。

当一个via头域的期待过来时，根据期待中的原始IP和端口号（其实就是存储在父连接跟踪的via头域IP地址和端口）建立NAT规则，将响应消息正确的转到内网的主机。将消息转给父连接来处理，父连接只需要根据存储在它这里的via和contact头域进行修改。、

Contact头域的期待处理与via头域一样。

* + 1. OutBound Response处理流程

处理过程

1. 分析Cseq头域，确定这个响应消息对应哪个请求。
2. Invite请求，获取响应消息码，如果为200表示session建立成功，为SDP中的RTP，RTCP建立期待。如果为100-199临时响应，这时还不能把确定session是否建立成功，但是这些临时响应可能携带SDP媒体信息，需要为它建立期待。如果为其他消息码说明session建立失败，删除SDP媒体信息期待。
3. 其他方法，如果携带SDP，就为它建立期待。
4. 根据配置对不支持的方法决定是否丢掉。

图5 - 5 OutBound Response处理流程

* + 1. 注册请求信息

注册请求消息

1. 根据请求消息中的via头域，建立期待，之所以要这么做事因为。SIP的客户端可能从一个端口发送请求，而从另外一个端口接收响应。修改via头域IP为期待结构外网IP，修改端口号为期待结构的外网端口。
2. Contact头域对于注册请求有特殊的含义，表示当前客户端想要组成的位置。为了能使外网用户呼叫内网用户，需要对contact头域建立期待（via头域和contact头域大多是不一样的，如果一样的就不需要再建立期待），保证注册服务器可以通过这个期待发送消息到注册这个地址用户。修改Contact头域为期待的外网IP和端口。
3. 因为是从内网发到外网，所以要修改SIP的头域。

注册响应消息

1. 使用头域Cseq匹配前面的请求时记录的Cseq，如有记录继续，否则直接修改这个SIP响应。
2. 若有记录进一步判断注册响应消息码是否为200成功响应。若是表示注册成功，刷新Contact头域期待结构的定时器，防止超时删除。若是表示注册失败，直接删除via和contact的头域期待信息。
3. 修改SIP的头域，将注册响应消息转到发出这个注册请求内网主机。



图5 - 6注册响应消息

* 1. 软件架构
     1. Linux netfilter简介

netfilter是内核用于提供各种网络服务的底层模块，netfilter框架中有很重要的5个点，每个点都可以挂载相应的hook对数据进行处理。这几个点分别是PREROUTING, FORWAD, POSTROUTING, INPUT, OUTPUT。

当一个数据包进入设备时，刚刚做完版本号和检验和的检验后，就会进入PREROUTING点，之后进入路由模块儿进行查路由，并判断是否是发往本机的数据包，如果是就进入INPUT点，之后会把数据包转给本机相应的进程处理，否则是路由转发，就进入FORWARD点。转发的数据包在FORWARD点和本机发出的数据包在OUTPUT点处理完之后都会转到POSTROUTING点进行处理，最后发到网络中去。见下图

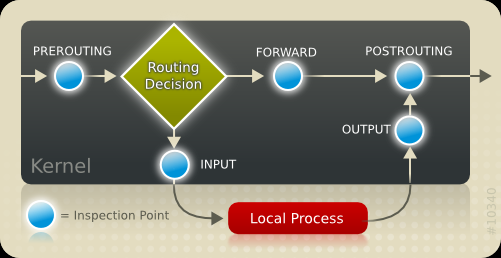


图5 - 7

Netfilter是一个很容易扩展的框架，在每个点上都可以挂载相应的钩子，对数据包进行处理，只有当一个点上的所有钩子都处理完之后才会转向下一个点。在不同的点上挂载上相应的钩子就能够实现不同的网络服务。

由于SIP-ALG需要做应用层的NAT，所以SIP-ALG模块是在POSTROUTING点上进行的网络服务扩展。

* + 1. 软件架构图

当设备从网络上接受到一个数据包时，netfilter把数据包从IPv4协议栈中勾起来，应用识别模块识别出是SIP数据包时，并转交给SIP-ALG模块处理。

IPv4协议栈是内核提供的，再此不做解释。

应用识别模块是用于识别各种网络应用，并把各种网络转换成一个唯一的crc值，再次基础之上可以做应用控制，当应用识别模块识别是SIP应用时，就会把数据包转交ALG- SIP模块，ALG-SIP模块在netfilter模块和conntrack模块基础之上，建立起应用层的NAT规则。

ALG-SIP模块，在内网的SIP话机向SIP代理服务器注册的时候建立期待，并把SIP数据包中包含IP和端口的信息修改成外网的IP和端口，当外网话机呼叫内网话机或内网话机 外网话机时就可以建立正确的连接。在发起Invite请求时SIP数据包中会携带SDP信息，ALG-SIP模块会根据SDP信息建立媒体信息期待，当收到200消息时如果携带SDP信息ALG-SIP也会建立媒体信息期待。因为建立了媒体信息的期待，当双方话机接通后媒体信息就能够沿着真确的数据连接通路实现媒体信息的传输。

配置下发模块是用户在UI上进行配置的ALG-SIP的控制信息，控制ALG-SIP的端口，对不支持请求方法的处理情况等各种配置信息，当用户配置完后，点击保存的时候，UI把数据传送到CGI，CGI程序处理完UI上的数据后并保存到相应的配置文件中，并调用配置下发程序进行配置下发，配置下发程序会读相关取配置文件把用户的配置下发到ALG-SIP模块，以便ALG-SIP模块能够根据用户的实际配置进行数据包处理。

软件架构如下图



图5 - 8

* 1. 测试数据

由于防火墙需要在各种环境下部署，所以SIP需要在NAT模式、路由模式、网桥模式、虚拟网线模式下进行穿透，下面是在NAT模式下进行的抓包分析的结果。

* + 1. NAT模式

由于SIP业务较多且其中有些业务比较复杂，下面从简单的业务开始测试，逐一到复杂的业务。

1. 话机注册

部署方式



图5 - 9

注册的信令流程



图5 - 10

当话机向proxy注册时,就会发送Register请求，用以验证用户的合法性，当proxy第一次收到话机的Register请求时，就会返回一个401的质问信息，当话机受到401的信息后，会再次发起Register请求并携带用户的验证信息，当proxy收到了第二次发起的Register请求时，如果用户验证通过则返回200 OK的响应给话机，并记录话机的IP和端口用以进行SIP信令交互。

信令交互



图5 - 11

当内网话机(假设Phone1)向Proxy注册时,Register请求数据包经过防火墙时根据Contact头域建立期待信息，并修改Contact头域的IP和端口为外网的IP和端口，并把数据包转发给Proxy，当Proxy收到Register请求时，会发送401质询信息，401质询信息经过防火墙会直接转发给Phone1，Phone1收到质询信息后会携带用户的验证信息再次发起Register请求，Register再次经过防火墙时不会再次建立期待，因为前面已经在第一次发起Register请求时已经建立过了，所以会根据之前建立起来的NAT规则修改Register请求中的Contact头域中的IP和端口后，转发给Proxy，当Proxy再次收到Register请求时，会根据Phone1的用户信息进行身份验证，验证通过后就会发起200 OK的响应，防火墙收到200 OK的响应后就转发给Phone1，此时就实现了话机经过ALG-SIP的注册。（以下Phone1简称P1）

表5 - 2

|  |  |  |  |
| --- | --- | --- | --- |
| 话机 | | AF | Proxy |
| P1 -> Proxy | Register  Contact :  Via :  192 . 168 . 88 . 100 : 5060 | Register  Contact :  Via :  200 . 200 . 155 . 167 : 5060 |
| Proxy -> P1 | 401  Via :  192 . 168 . 88 . 100 : 5060 | 401  Via :  200 . 200 . 155 . 167 : 5060 |
| P1 -> Proxy | Register  Contact :  Via :  192 . 168 . 88 . 100 : 5060 | Register  Contact :  Via :  200 . 200 . 155 . 167 : 5060 |
| Proxy -> P1 | 200 OK  Contact :  192 . 168 . 88 . 100 : 5060 | 200 OK  Contact :  200 . 200 . 155 . 167 : 5060 |

建立的期待

表5 - 3

|  |  |
| --- | --- |
| DestIP-Port | SaveIP-Port |
| 200 . 200 . 155 . 170 : 5060 | 192 . 168 . 88 . 100 : 5060 |

在次P1完成了注册后,会在防火墙中留下如上的期待信息,并且Proxy会保存200 . 200 . 155 . 167 : 5060作为与P1进行SIP信令交互的IP和端口,当Proxy向P1发起SIP请求或是响应时就会将SIP数据包发到200 . 200 . 155 . 167 : 5060上去,此时防火墙会根据注册时留下的期待信息,将该数据包转发到P1,并且话机在注册的时候在Expires头域中表明了注册超时时间一般为3600秒,当超时时间一到防火墙的ALG模块会删掉之前建立的SIP信令交互的期待,等待P1话机的再次注册，而话机P1会每隔3600秒之后进行重新注册。

1. 两台话机在内网，代理服务器在外网

部署方式



图5 - 12

信令交互流程

由于SIP数据包在交互的过程中，有些数据不包含SDP信息，对于建立连接也不起作用（小于200的响应），在分析数据包的时候不进行分析，抓住主要的数据包进行分析。

表5 - 4

|  |  |  |  |
| --- | --- | --- | --- |
| 话机 | | AF | Proxy |
| P1 -> Proxy | Invite  SDP C :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | Invite  SDP C :  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| Proxy -> P3 | Invite  SDP C :  200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 10006 | Invite  SDP C :  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| P3 -> Proxy | 200 OK  SDP C :  192 . 168 . 88 . 103 : 3336  192 . 168 . 88 . 103 : 10006 | 200 OK  SDP C :  200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 10008 |
| Proxy -> P1 | 200 OK  SDP C :  200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 10010 | 200 OK  SDP C :  200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 10008 |

在上表中请求，当内网话机P1呼叫P3时，P1首先Proxy发起Invite请求，当Proxy收到Invite请求数据包后，会根据P3话机注册时留下的信息，把Invite请求转发给P3，当P3收到Invite请求时P3就知道P1的IP地址和端口，之后P3会向Proxy发送200 OK的响应，Proxy收到后会把P3发来的响应转发给P1，当P1收到Proxy转发的200 OK的响应时，P1就知道了P3的IP地址和端口，之后P1就会发送ACK给Proxy，Proxy会把ACK响应转发给P3，当P3收到ACK后，P1和P3进行通话的时要进行媒体传输的IP地址和端口就动态的协商好了。防火墙在SIP数据包经过的时候就会为携带有SDP信息的数据包建立媒体信息期待，并且根据NAT规则进行IP地址和端口的转换，保证P1和P3通话正常进行。在此过程中建立的期待如下表

表5 - 5

|  |  |  |
| --- | --- | --- |
| 方向 | DestIP-Port | SaveIP-Port |
| Invite  P1 -> Proxy | 200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 3335  200 . 200 . 155 . 167 : 10004  200 . 200 . 155 . 167 : 10005 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| Invite  Proxy -> P3 | 200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 3337  200 . 200 . 155 . 167 : 10006  200 . 200 . 155 . 167 : 10007 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 200 OK  P3 -> Proxy | 200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 3339  200 . 200 . 155 . 167 : 10008  200 . 200 . 155 . 167 : 10009 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |
| 200 OK  Proxy -> P1 | 200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 3341  200 . 200 . 155 . 167 : 10010  200 . 200 . 155 . 167 : 10011 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |

当P1向Proxy发送Invite请求时，数据包经过防火墙发到外网时，会建立一个期待，即当收到目的IP200 . 200 . 155 . 167为和端口33334的数据包时，就按SaveIP-Port的IP地址和端口进行转发，即会把数据包转发到P1，当数据包从防火墙发出去的时候携带的SDP信息已经被ALG-SIP模块修改成外网的IP和端口（200 . 200 . 155 . 167 : 3334和200 . 200 . 155 . 167 : 10004），当Proxy收到该Invite数据包后，就会根据P3注册时留下的IP地址和端口进行转发，此时这个数据包会再次经过防火墙发到内网时会再次建立期待（200 . 200 . 155 . 167 : 3336 10006指向200 . 200 . 155 . 167 : 3334 10004），在建立期待时首先会根据目的IP和端口查找期待表，找到后会根据查找到后的期待对新建立的期待进行修正（修正后的结果是200 . 200 . 155 . 167 : 3336 10006指向192 . 168 . 88 . 100 : 3334 10004），否则就会使期待指向防火墙自己，导致无法建立正常的通话。P3收到Proxy转发发到防火墙，经防火墙修改后的数据包后，会回复200 OK的响应并携带P3自己的SDP信息，当200 OK相应进过防火墙时,防火墙会为200 OK相应建立期待并修改其携带的SDP信息中的IP地址和端口，当Proxy收到P3发的200 OK相应时，会根据P1在注册时留下的IP地址和端口进行转发200 OK的响应，当Proxy发到防火墙的200 OK的响应时，会根据目的IP和端口在期待表中进行查找，找到后会根据找到的期待对新的期待进行修正，之后会修改SDP信息，并发给P1。P1和P3交互完成后，其进行通信的IP地址和端口就协商出来了，P1和P2就能进行整的通话和视频业务。如下图是P1和P2通话的数据流向

表5 - 6

|  |  |  |
| --- | --- | --- |
| P1 | AF | P3 |
|  | | |
| Src :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004  Dest :  200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 10010 | Src :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004  Dest :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004 | Src :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004  Dest :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004 |
|  | | |
| Src :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004  Dest :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | Src :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004  Dest :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | Src :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004  Dest :  200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 10006 |

当P1向P2发送媒体信息时,就会把媒体数据包,发送到200 . 200 . 155 . 167 : 3340(音频)和200 . 200 . 155 . 167 : 10010(视屏),防火墙会根据建立起来的NAT规则进行转发,并修改目的IP和端口,并转发给P3;当P3要向P1发送数据时会把数据发往200 . 200 . 155 . 167 : 3336(音频)和200 . 200 . 155 . 167 : 10006(视屏),防火墙会根据建立起来的NAT规则进行转发,并修改目的IP和端口,并转发给P1，这样P1和P3之间就能够正常通话和视屏。

1. 特权业务

业务介绍

假设Phone2具有特权业务，当Phone1和Phone3正在通话时，Phone2拨打Phone1的号码，Phone2可以直接和Phone1建立通话，而Phone3的通话被保留，当Phone2与Phone1结束通话后（Phone2挂机后），Phone1和Phone3还能继续通话。

部署方式



图5 - 13

信令交互流程

在进行交互的过程中有些对数据包对会话的建立并不起太大的作用，所以忽略其中次要的数据包，抓住主要的数据包进行分析。如图

首先需要在Proxy端为Phone2添加特权业务权限，然后进行测试。特权业务分为三个部分，其中每个部分在图中我已经用红色的粗线条标明出来。

第一部分：Phone1和Phone3通话过程，在图中的信令交互分别是F1-F8，当然其中还有些如trying，ring，ack等请求和相应，跟分析该业务没有太大的关系所以在图中忽略，想知道详细的交互流程请参考第二章。

当Phone1向Phone2发起呼叫时，Phone1的Invite请求经过防火墙时，ALG-SIP会根据其携带的SDP信息为Phone1建立媒体信息期待，并修改Invite请求的中SDP的IP和端口为外网的IP和端口，当Proxy收到Phone1发送来的Invite请求后，会记录下Phone1的SDP信息，并把Invite请求原封不动的转发给防火墙，防火墙中ALG-SIP会根据其中的SDP信息到期待表中进行查找，找到后会根据已有的期待信息对新建立的媒体期待进行修正使其指向Phone1，然后把Invite请求转发给Phone3，Phone3收到Phone1的Invite请求后就知道了要与Phone1进行通话的IP地址和端口，此时协商只进行了一半；Phone3收到Invite请求后会向Proxy回复一个200 OK的响应，当200 OK的响应经过防火墙时，ALG-SIP会为Phone3建



图5 - 14

立媒体期待信息，并修改200 OK中SDP信息中的IP何端口为外网的IP和端口和转发给Proxy，当Proxy收到200 OK响应后会记录下Phone3的媒体信息，并把200 OK的响应原封不动转发给Phone1，200 OK经过防火墙的时候，再次建立期待，并根据200 OK中SDP信息进行查找期待表，根据找到的期待信息进行对新建立的期待进行修正使其指向Phone3，然后防火墙把200 OK转发给Phone1，此时Phone1就知道了Phone3的媒体信息的IP和端口，在Phone1收到200 OK后会发出一个ACK确认给Phone3，经过防火墙到达Proxy在经过防火墙到达Phone3，这样一个通话的信令交互过程就完成了，并且双发都协商出了进行通信的IP和端口。

第二部分：Phone2和Phone1的通话过程，Phone3的通话被保持，其中的信令交互过程为F9-F20，当然其中还有些如trying，ring，ack等请求和相应，跟分析该业务没有太大的关系所以在图中忽略，想知道详细的交互流程请参考第二章。

当Phone1和phone3建立通话之后，拥有特权业务的Phone2此时正想给Phone1打电话时，Phone2就会直接和Phone1进行通话，首先Phone2摘机拨打Phone1的号码会发送Invite请求给Phone1，当Invite请求经过防火墙时，ALG-SIP会为Phone2的媒体信息建立期待，并修改SDP中的IP和端口为外网的IP和端口和转发给Proxy，当Proxy收到Invite请求后会记录下Phone2的媒体信息并转发Invite请求给Phone1，当Invite请求经过防火墙时，ALG-SIP会根据其携带的SDP信息进行查找期待信息，找到后会根据查到的期待对新建立的期待进行修正使之指向Phone2并把Invite请求转发给Phone1，当Phone1收到Invite请求后会回复一个200 OK的响应，200 OK的响应经过防火墙时因为已经建立了期待，所以不需要再建立期待，防火墙修改其SDP中的信息之后转发给Proxy，Proxy收到后转发给Phone2，当200 OK经过防火墙时因为已经建立了期待，所以不需要再建立期待然后转发给Phone2，Phone2收到后发出ACK的确认信息，ACK经过防火墙到达Proxy再经过防火墙到达Phone1，此时Phone2和Phone1的信令交互过程已经完成了，双方也协商出了通话的IP和端口。

在Invite F11的同时Proxy还需要对Phone3通话进行保持，所以Proxy向Phone3发起了Invite F13的请求其中的SDP信息是让Phone3听忙音的音频信息，当Invite请求经过防火墙时会建立新的期待并转发Invite请求给Phone3，当Phone3收到Invite请求后回复200 OK响应，其中的SDP信息只携带了音频的媒体信息，当200 OK经过防火墙时，建立新的期待并修改200 OK中的SDP信息中的IP和端口为外网IP和端口然后转发给Proxy，Proxy收到200 OK响应后就开始给Phone3播放等待的忙音，Phone3就处于等待状态。

第三部分：当Phone2和Phone1通话结束后，Phone2挂机，Phone1和Phone3还能进行正常的通话，其中信令的交互过程为F21-F32，当然其中还有些如trying，ring，ack等请求和相应，跟分析该业务没有太大的关系所以在图中忽略，想知道详细的交互流程请参考第二章。

当Phone2和Phone1通话结束后，Phone2挂机时会发起Bye请求，Bye请求经过防火墙并转发给Proxy，Proxy收到Bye请求后会回复200 OK的响应，200 OK的响应经过防火墙然后转发给Phone2，此时Phone2的通话已经结束，此时Proxy再次向Phone1发起Invite请求，Invite请求中携带的SDP信息是Phone3在Proxy端最后一次留下的媒体信息，当Invite经过防火墙时会建立期待，并根据SDP信息查找期待表找到后根据找到的期待信息对新的期待信息进行修正使之指向Phone3然后转发给Phone1，当Phone1收到Invite请求后会回复一个200 OK的响应给Proxy，当200 OK经过防火墙时建立期待并修改200 OK中的SDP信息中的IP和端口为外网的IP和端口然后转发给Proxy，当Proxy收到200 OK的响应后，会向Phone1发送ACK响应，然后向Phone3发起Invite请求，Invite请求中携带的SDP信息是刚刚收到的Phone1回复的200 OK中的SDP信息，当Invite信息经过防火墙时，建立期待并查找期待表用查到的期待信息修正新的期待信息使之指向Phone1让后转发给Phone3，当Phone3收到Invite请求后，会恢复一个200 OK给Proxy，当200 OK经过防火墙时会建立期待并使之指向Phone3，当Proxy收到200 OK的响应后会回复ACK响应给Phone3，到此Phone1和Phone3的通话已经恢复，但是在此有一个关键的地方时当Proxy向Phone1发起Invite请求的时候是模拟Phone3发起的并且Invite请求中的SDP信息也是老的SDP信息，所以ALG-SIP不能够删除之前Phone3建立的期待，否则到Invite请求经过防火墙时就查找不到期待，从而导致新建立的期待是指向防火墙自己，从而导致双方无法正常通话和视屏。

在此三步过程中数据包如下表

表5 - 7

|  |  |  |  |
| --- | --- | --- | --- |
| 话机 | | AF | Proxy |
| Invite  Phone1 -> Prxoy | Invite SDP C：  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | Invite SDP C：  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| Invite  Proxy -> Phone3 | Invite SDP C：  200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 10006 | Invite SDP C：  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| 200 OK  Phone3 -> Proxy | 200 OK  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004 | 200 OK  200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 10008 |
| 200 OK  Proxy -> Phone1 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 10010 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 10008 |
| Invite  Phone2 -> Proxy | Invite SDP C :  192 . 168 . 88 . 102 : 3334  192 . 168 . 88 . 102 : 10004 | Invite SDP C :  200 . 200 . 155 . 167 : 3342  200 . 200 . 155 . 167 : 10012 |
| Invite  Proxy -> Phone1 | Invite SDP C :  200 . 200 . 155 . 167 : 3344  200 . 200 . 155 . 167 : 10014 | Invite SDP C :  200 . 200 . 155 . 167 : 3342  200 . 200 . 155 . 167 : 10012 |
| Invite  Proxy -> Phone3 | Invite SDP C :  200 . 200 . 155 . 170 : 6014 | Invite SDP C :  200 . 200 . 155 . 170 : 6014 |
| 200 OK  Phone3 -> Proxy | 200 OK SDP C :  192 . 168 . 88 . 103 : 3334 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3346 |
| 200 OK  Phone1 -> Proxy | 200 OK SDP C :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3348  200 . 200 . 155 . 167 : 10016 |
| 200 OK  Proxy -> Phone2 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3350  200 . 200 . 155 . 167 : 10018 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3348  200 . 200 . 155 . 167 : 10016 |
| Invite  Proxy -> Phone1 | Invite SDP C :  200 . 200 . 155 . 167 : 3352  200 . 200 . 155 . 167 : 10020 | Invite SDP C :  200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 10008 |
| 200 OK  Phone1 -> Proxy | 200 OK SDP C :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3354  200 . 200 . 155 . 167 : 10022 |
| Invite  Proxy -> Phone3 | Invite SDP C :  200 . 200 . 155 . 167 : 3356  200 . 200 . 155 . 167 : 10024 | Invite SDP C :  200 . 200 . 155 . 167 : 3354  200 . 200 . 155 . 167 : 10022 |
| 200 OK  Phone3 -> Proxy | 200 OK SDP C :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3358  200 . 200 . 155 . 167 : 10026 |

在此三步过程中AF建立的期待如下表

表5 - 8

|  |  |  |
| --- | --- | --- |
| 方向 | DestIP-Port | SaveIP-Port |
| Invite  Phone1 -> Prxoy | 200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 3335  200 . 200 . 155 . 167 : 10004  200 . 200 . 155 . 167 : 10005 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| Invite  Proxy -> Phone3 | 200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 3337  200 . 200 . 155 . 167 : 10006  200 . 200 . 155 . 167 : 10007 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 200 OK  Phone3 -> Proxy | 200 . 200 . 155 . 167 : 3338  200 . 200 . 155 . 167 : 3339  200 . 200 . 155 . 167 : 10008  200 . 200 . 155 . 167 : 10009 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |
| 200 OK  Proxy -> Phone1 | 200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 3341  200 . 200 . 155 . 167 : 10010  200 . 200 . 155 . 167 : 10011 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |
| Invite  Phone2 -> Proxy | 200 . 200 . 155 . 167 : 3342  200 . 200 . 155 . 167 : 3343  200 . 200 . 155 . 167 : 10012  200 . 200 . 155 . 167 : 10013 | 192 . 168 . 88 . 102 : 3334  192 . 168 . 88 . 102 : 3335  192 . 168 . 88 . 102 : 10004  192 . 168 . 88 . 102 : 10005 |
| Invite  Proxy -> Phone1 | 200 . 200 . 155 . 167 : 3344  200 . 200 . 155 . 167 : 3345  200 . 200 . 155 . 167 : 10014  200 . 200 . 155 . 167 : 10015 | 192 . 168 . 88 . 102 : 3334  192 . 168 . 88 . 102 : 3335  192 . 168 . 88 . 102 : 10004  192 . 168 . 88 . 102 : 10005 |
| Invite  Proxy -> Phone3 | 200 . 200 . 155 . 170 : 6014  200 . 200 . 155 . 170 : 6015 | 200 . 200 . 155 . 170 : 6014  200 . 200 . 155 . 170 : 6015 |
| 200 OK  Phone3 -> Proxy | 200 . 200 . 155 . 167 : 3346  200 . 200 . 155 . 167 : 3347 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335 |
| 200 OK  Phone1 -> Proxy | 200 . 200 . 155 . 167 : 3348  200 . 200 . 155 . 167 : 3349  200 . 200 . 155 . 167 : 10016  200 . 200 . 155 . 167 : 10017 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 200 OK  Proxy -> Phone2 | 200 . 200 . 155 . 167 : 3350  200 . 200 . 155 . 167 : 3351  200 . 200 . 155 . 167 : 10018  200 . 200 . 155 . 167 : 10019 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| Invite  Proxy -> Phone1 | 200 . 200 . 155 . 167 : 3352  200 . 200 . 155 . 167 : 3353  200 . 200 . 155 . 167 : 10020  200 . 200 . 155 . 167 : 10021 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |
| 200 OK  Phone1 -> Proxy | 200 . 200 . 155 . 167 : 3354  200 . 200 . 155 . 167 : 3355  200 . 200 . 155 . 167 : 10022  200 . 200 . 155 . 167 : 10023 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| Invite  Proxy -> Phone3 | 200 . 200 . 155 . 167 : 3356  200 . 200 . 155 . 167 : 3357  200 . 200 . 155 . 167 : 10024  200 . 200 . 155 . 167 : 10025 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 200 OK  Phone3 -> Proxy | 200 . 200 . 155 . 167 : 3358  200 . 200 . 155 . 167 : 3359  200 . 200 . 155 . 167 : 10026  200 . 200 . 155 . 167 : 10027 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3335  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |

1. 呼叫前传业务

业务介绍

呼叫前传业务分为四种，分别是无条件呼叫前传、遇忙呼叫前传、无应答呼叫前传和离线呼叫前传，由于三种业务的信令交互都很相似，所以在此只介绍一种最简单的无条件呼叫前传业务，无条件呼叫前传业务在使用时需要在Proxy端为话机添加权限，话机申请后就可以使用，在此假设Phone2具有无条件呼叫前传业务的权限，当Phone1打Phone2的时候，Phone1的通话被无条件的转接到Phone3，Phone3摘机后建立与Phone1的通话。

部署方式图



图5 - 15

信令交互图

在进行交互的过程中有些对数据包对会话的建立并不起太大的作用，所以忽略其中次要的数据包，抓住主要的数据包进行分析。如图

首先需要在Proxy端给Phone2添加无条件呼叫前转移业务权限，然后再Phone2端申请该业务，会听到您申请的新业务以登记欢迎使用的提示音后，表示业务申请成。



图5 - 16

该业务分为两个部分，在途中我已经用红色的线条标记出来了，第一部分为呼叫Phone2，由于Phone2具有无条件呼叫前传业务，所以Phone1的呼叫被保留，Phone1端听等待忙音。第二部分为转接到Phone3，当Phone3摘机后于Phone1建立通话，业务完成。

第一部分：首先Phone1拨打Phone2号码，Phone1发起Invite请求给Phone2，当Invite F1请求到达防火墙时，防火墙为Invite F1建立媒体信息期待，并修改Invite F1中的SDP信息中的IP地址和端口为外网的IP地址和端口然后把Invite F2转发出去，当Proxy收到Invite F2时会记录下Invite F2中的SDP信息，由于Phone2具有无条件呼叫转移业务，所以Invite F2不会转发出去，Proxy此时采取的动作时保持Phone1的呼叫即向Phone1恢复一个183 F3当防火墙收到183 F3时会建立媒体信息的期待，由于183 F3里面只有音频的信息所以建立的期待只有音频信息的期待，然后防火墙转发183 F4给Phone1，此时Phone1就开始听忙音，Phone1听到的声音是“正在转接中请稍后”。、

第二部分：当Proxy给Phone1发送183的响应后，由于Phone2被无条件转接到Phone3，所以会立即发起一个Invite F5（其中的SDP信息是Phone1在Proxy端留下的信息）请求给Phone3，当防火墙收到Invite F5后会建立媒体期待，并查找防火墙的媒体期待信息根据找打的媒体期待信息对新建立的期待进行修正使之只想Phone1，然后把Invite F6转发给Phone3，当Phone3收到Invite F6后会回复一个200 OK F7给Phone1，当200 OK F7经过防火墙是会建立指向Phone3的媒体信息期待并修改200 OK F7的SDP信息中的IP地址和端口为外网的IP地址和端口，然后转发200 OK F8给Proxy，当Proxy收到200 OK F8后会记录下Phone3的媒体信息，然后转发200 OK F9给Phone1，当200 OK F9经过防火墙是，会建立新的期待，并查找防火墙中的媒体信息期待，找到后根据找到的媒体信息期待修正新建立的媒体信息期待使之指向Phone3，然后转发200 OK F10给Phone1，当Phone1收到200 OK F10后会恢复一个ACK的确认信息，ACK经过防火墙到Proxy，再经Proxy转发过防火墙到Phone3，到此Phone1和Phone3协商已经完成，双方的通话的IP地址和端口也已经协商出来了，然后就是Phone1和Phone3分别向Proxy发送一个INFO的请求表示双发都已经做好了通话的准备。然后Phone1和Phone3就可以进行正常通话和视频了。

数据包分析

表5 - 9

|  |  |  |  |
| --- | --- | --- | --- |
| 话机 | | AF | Proxy |
| Invite  Phone1 -> Proxy | Invite SDP C :  192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 10004 | Invite SDP C :  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| 183  Proxy -> Phone1 | 186  200 . 200 . 155 . 170 : 6090 | 183  200 . 200 . 155 . 170 : 6090 |
| Invite  Proxy -> Phone3 | Invite SDP C :  200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 10006 | Invite SDP C :  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004 |
| 200 OK  Phone3 -> Proxy | 200 OK SDP C :  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 10010 |
| 200 OK  Proxy -> Phone1 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3342  200 . 200 . 155 . 167 : 10012 | 200 OK SDP C :  200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 10010 |

媒体期待表

表5 - 10

|  |  |  |
| --- | --- | --- |
| 方向 | DestIP-Port | SaveIP-Port |
| Invite  Phone1 -> Proxy | 200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 3334  200 . 200 . 155 . 167 : 10004  200 . 200 . 155 . 167 : 10005 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 183  Proxy -> Phone1 | 200 . 200 . 155 . 170 : 6090  200 . 200 . 155 . 170 : 6091 | 200 . 200 . 155 . 170 : 6090  200 . 200 . 155 . 170 : 6091 |
| Invite  Proxy -> Phone3 | 200 . 200 . 155 . 167 : 3336  200 . 200 . 155 . 167 : 3337  200 . 200 . 155 . 167 : 10006  200 . 200 . 155 . 167 : 10007 | 192 . 168 . 88 . 100 : 3334  192 . 168 . 88 . 100 : 3335  192 . 168 . 88 . 100 : 10004  192 . 168 . 88 . 100 : 10005 |
| 200 OK  Phone3 -> Proxy | 200 . 200 . 155 . 167 : 3340  200 . 200 . 155 . 167 : 3341  200 . 200 . 155 . 167 : 10010  200 . 200 . 155 . 167 : 10011 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |
| 200 OK  Proxy -> Phone1 | 200 . 200 . 155 . 167 : 3342  200 . 200 . 155 . 167 : 3343  200 . 200 . 155 . 167 : 10012  200 . 200 . 155 . 167 : 10013 | 192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 3334  192 . 168 . 88 . 103 : 10004  192 . 168 . 88 . 103 : 10005 |

1. 结论与展望
   1. 结论

本文中采用了ALG的实现方案解决了SIP穿透NAT/Firewall，把防火墙部署在私网和公网的边界上，并配置成NAT模式进行下，进行各种业务的语音和视屏通话测试，验证了方案是可行的，整个系统运行比较稳定。

本文对采用ALG方案解决SIP穿透NAT问题的设计和实现方案做了详细的介绍，对处于NAT部署模式下的SIP穿透防火墙的各种业务的SIP信令交互做了详细的分析，对防火墙内部处理SIP请求信令并结合经过防火墙的数据包进行了详细的分析并给出了实现数据。

* 1. 展望

VoIP 技术作为三网融合的一个重要支撑，一直都受到业界的关注。对于第一代 VoIP 技术的 H.323 协议，虽然还能从其实现上看到电话信令的身影，但是确实已经向IP网承载多媒体业务跨出了坚实的一步，而且也作为网络电话通信的主流协议被广泛的使用。

SIP协议作为NGN通信的核心协议将有着极大的市场潜力和应用前景。SIP协议是通信的基础，尤其是在3G和VoIP中，SIP的灵活性和可扩展性都将得到体现并受到人们的欢迎。随着 SIP 扩展协议对SIP核心的逐渐完善和补充，SIP标准将发挥越来越重要作用，可以预见在不远的将来，SIP协议逐渐占领着VoIP技术领域，慢慢走向成熟，开始动摇 H.323协议族的主导地位，，在下一代网络NGN和软交换中，SIP也已经成为核心协议之一。

由于现有的NAT/Firewall部署结构大量的存在，阻碍了VoIP技术的发展，采用ALG方案解决了SIP穿透NAT/Firewall的问题，为VoIP技术向企业网和城域网的发展扫清了障碍，这无疑具有很大的市场潜力和发展前景。

参考文献

1. M.Handley, H.Schulzrirme, E.Schooler, J.Rosenberg，SIP:Session initiation ProtocoI, RFC2543, March 1999
2. 张连靖，基于Linux的SIP穿越NAT研究与实现，2004年5
3. RFC2327文档中文版
4. 高磊，基于ALG的VoIP业务穿透NAT的研究与设计，2007年1月
5. 严军，NGN网络业务NAT穿越技术探讨，通讯世界，2003
6. 吴伟，NGN业务穿越NAT/FW的解决方案，2004年12月
7. 旷志荣，王芙蓉，基于MIDCOM 协议的SIP 穿越NAT/FW，2005年5月6
8. 郭常清，针对SIP的STUN解决方案的设计与实现，2005年6月
9. Audio-Video Transport Working Group, H.Schulzrinne, S.Casner, R. Frederick, VJacobson. RTP:A Transport Protocol for Real-Time Applications. RFC 1889 January 1996