华中科技大学

硕士学位论文

一种利用TURN穿越对称型NAT方案的设计与实现

姓名：闵江

申请学位级别：硕士

专业：通信与信息系统

指导教师：杜旭

20080528

摘 要

为了在 IPv6 广泛投入使用之前缓解 IPv4 地址空间短缺的问题，NAT 技术被提出。一些标准的应用协议和通用标准协议，同 NAT 配合工作得很好。然而有些服务依赖于端到端的数据包，例如 SIP 消息体中包含后继的 RTP 媒体流互通所需要的 IP 以及端口信息，这样就会使得使用 SIP 作为信令传输的网络应用程序在经过 NAT 后无法正常工作。因此，对于 NAT 的穿越是迫切需要解决的问题。

而现有的几种穿越方式都有其局限性。如 ALG 方式要求现有的 NAT 设备需升级支持相应协议，STUN 则无法穿越对称型 NAT。关于对称型 NAT 的穿越，业界既无标准协议可循也没有成熟的技术可供参考，这使得对对称型 NAT 的穿越，成为当今的一个难点。

本文针对在 IP 上承载语音和视频的协议的控制通道/媒体通道无法穿越对称 NAT 与公网进行互通的问题，设计并完成对称 NAT 的一种穿越方式，主要使用 TURN 技术。利用 TURN 服务器的中转，TURN 客户端穿越对称 NAT，与外部主机进行正常通信。另外，本文更提出了一种将 STUN 方式与 TURN 方式相结合的 NAT 穿越方案，既解决了所有类型 NAT（包括对称型 NAT）的穿越问题，又降低了 TURN 服务器的负荷，同时也无需 NAT 的任何修改。与传统穿越技术相比，其最大的特点是不仅可以穿越一般类型的 NAT，而且可以穿越安全系数很高的对称性的 NAT。该方案已成功应用于基于嵌入式 Linux 平台的 VoIP 网关中。

关键词：网络地址转换; 对称型 NAT; TURN; 基于 IP 的语音; 会话初始协议

Abstract

In order to resolve the problem of the shortage of IPv4 addresses before IPv6 widely used, NAT technology has been proposed. Some standard application protocols work well with NAT. However, some end-to-end services rely on data packets, such as SIP in which the message body contains the IP and port information needed by the following RTP media stream. This will cause the network application programs using SIP as the signaling not to work properly. Therefore, the NAT traversing problem is the one which needs to be solved urgently.

There exists several NAT traversing methods, but each has its limitation. For example, ALG requires that the existing NAT equipments must support the corresponding software update. The STUN method can not traverse symmetric NAT. For the traversing of symmetric NAT, there are neither standard protocols nor mature technologies. This makes the traversing of symmetric NAT to be a difficult today.

The control/media channel carrying voice/video protocols based on IP faces the challenge in traversing symmetric NAT. This issue deals with this problem, and introduces a TURN based solution. The TURN client traverses through the NAT to communicate with external host by the forwarding of TURN server. With the combination of STUN, this scheme can not only traverse all types of NAT, but also reduce the load of TURN server. Besides, it dose not need any modification of the NATs. Compared with the traditional traversing methods, its greatest feature is that it can not only traverse the general types of NAT, but also traverse the symmetric NAT. This solution has been adopted successfully in the VoIP Gateway based on the embedded Linux platform.

Keyword： Network Address Translator (NAT); Symmetric NAT; Traversal Using Relay NAT (TURN); Voice over Internet Protocol (VoIP); Session Initiation Protocol (SIP)

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本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和汇编本学位论文。

保密□，在_____年解密后适用本授权数。

本论文属于

不保密□　。

（请在以上方框内打“ √ ”）

学位论文作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月　日

1 绪论

1.1 课题背景及意义

随着 Internet 用户的迅速增加，IPv4[1]地址资源面临耗尽的境地。为了在短期内解决 IPv4 地址匮乏的状况，NAT（Network Address Translator，网络地址翻译）[2]技术被提出。NAT 可以使私网内的多台 PC 机或设备共享单个、可全局路由的 IPv4 地址，同时可以阻止外网用户直接访问内部网络，增加内网的安全性。然而 NAT 的出现也给 VoIP（Voice over IP，基于 IP 的语音）带来问题。目前 VoIP 的主要标准有 H.323[3][4]和 SIP[5]（Session Initiation Protocol，会话初始化协议），SIP 因其简单、灵活和扩展性强等特点已经成为 VoIP 中的关键技术。由于协议自身设计上的原因，这些协议的控制通道/媒体通道难以穿越 NAT 设备。所以 NAT 的存在严重地制约了 VoIP 的应用。

现在已存在多种 NAT 穿越方式，但对于对称 NAT 的穿越却一直是一个难点。按照处理端口映射的方式以及与 NAT 外主机通信的方式上的不同，NAT 可以分为四种类型：完全透明型 NAT （Full Cone NAT），受限型 NAT （Restricted Cone NAT），端口受限型 NAT （Port Restricted Cone NAT）和对称型 NAT （Symmetric NAT）[6] [7]。前面三种 NAT，对来自同一内部 IP 地址和端口的所有请求，都会映射到同一外部 IP 地址和端口。而对于对称 NAT，即使是来自于同一内部 IP 地址和端口的数据包，目的 IP 地址和端口的不同，也会映射成不同的外部 IP 地址和端口。这会导致即使通过外部服务器获知了内部客户端在 NAT 上映射的地址和端口，然而因内部客户端与外部用户建立的是新的连接，即映射新的地址和端口，使得先前通过外部服务器获知的映射地址和端口变得毫无意义。

该课题的目的就是要解决在 IP 上承载语音和视频的协议的控制通道/媒体通道难以穿越对称 NAT 与公网进行互通的问题。

本文以与某企业合作的“基于嵌入式 Linux 的 VoIP 家庭网关”项目为背景，从基于 SIP 的 VoIP 家庭网关难以穿越对称 NAT 设备出发，设计并完成 TURN[8]（Traversal Using Relay NAT， 通过中继方式穿越 NAT）协议的关键部件 TURN 客户端和 TURN 服务器，在此基础上提出了一种基于 TURN 协议的 SIP 穿越 NAT 的方案，该方案满足了 VoIP 网关穿越 NAT 设备的需求，也为 TURN 协议与应用程序如

何进行结合提供了一种思路。该方案已成功应用于基于嵌入式 Linux 平台的 VoIP 网关中，并可推广到其它类型的终端上，为 VoIP 业务的推广应用扫清障碍。

1.2 国内外发展现状

为了解决 NAT 穿越问题，业界提供一些解决方案，主要有如下几种：

ALG （Application Level gateway，应用层网关）[9]方式：

该方式需要对现有防火墙/NAT 进行升级，应用层网关设计成能够识别指定 IP 协议（比如 H.323 和 SIP 协议）的防火墙。它不是简单地察看包头信息来决定数据包是否可以通过，而是更深层的分析数据包负载内的数据，也就是应用层的数据[10]。H.323 和 SIP 协议都在负载中放了重要的控制信息，例如语音和视频终端使用哪一个数据端口来接收对方终端的语音和视频数据。ALG 方式修改通过的信令，以保证 NAT 的穿越。因此，对应与不同的信令协议，都需要与之匹配的 ALG 方式。

UPnP （Universal Plug and Play，通用即插即用) [11]方式：

UPnP 是为了在电脑、智能家电和智能设备之间建立无所不在的网络连接而提出的协议体系[12]。由微软公司发起，在 1999 年成立了一个开放的产业联盟 UPnPForum，并制订了一系列标准。UPnP 允许客户端软件发现和配置包括 NAT 和防火墙在内的网络部件，其中后者需要安装 UPnP 软件。目前只有部分 NAT 和防火墙支持 UPnP 功能。

MIDCOM （Middle box Communications，中间盒通讯方式）[13]方式

MIDCOM 的目的是使用第三方应用程序(包括硬件设备)来控制防火墙和 NAT 设备并动态地决定安全策略，以此来适应 VoIP 业务[14]。第三方的应用程序使用 MIDCOM 定义的协议(或私有协议)控制防火墙和 NAT 设备，根据 VoIP 的需要动态地打开呼叫信令、媒体流互通的 IP 地址和端口号。

STUN （Simple Traversal of UDP Through NAT，UDP 对 NAT 的简单穿越方式）[15] [16]方式：

STUN 方式采用客户机/服务器模式。它允许应用程序探测当前在它们与公网之间是否存在 NAT、防火墙以及它们的类型，并且具备能够探测到 NAT 所分配的公网地址和端口的能力。STUN 协议中定义了两个实体：STUN 客户端和 STUN 服务器。STUN 客户端嵌入在终端系统的应用程序中，比如 SIP 用户代理，它向 STUN 服务器发送请求；STUN 服务器接收请求并产生 STUN 响应。SIP 终端在建立呼叫之前，通过向处在公网上的 STUN 服务器发送 STUN 请求，得到信令和媒体流在 NAT 上的

映射地址，并且将这些地址填写到 SIP 消息中的 Via、Contact 字段以及 SDP 中的媒体流传输地址，替代原有的私有地址。但是，STUN 只能工作在完全透明型 NAT、受限型 NAT 以及端口受限型 NAT 的网络环境下，在对称型 NAT 的情况下，SIP 用户代理通过 STUN 请求得到的映射地址是无效的。

TURN 方式：

TURN 协议在语法和操作上均与 STUN 相似，其优点是提供了对对称性 NAT 的穿越。处在公网的 TURN 服务器为客户端提供本身的一个外部 IP 地址和端口，并且负责中转通信双方的媒体流。

本文与现有技术的区别是，可以穿越对称型 NAT，而且设计出一种将 STUN 方式与 TURN 方式结合穿越 NAT 的方案。对于对称 NAT，使用 TURN 即通过中继方式进行穿越，利用 TURN 服务器的中转，TURN 客户端穿越对称 NAT，与外部主机进行正常通信；而与 STUN 结合，既可以穿越所有类型的 NAT，又可降低 TURN 服务器的负载。

1.3 课题创新点

现有的几种穿越方式都有其局限性。如 ALG 和 MIDCOM 方式要求现有的 NAT 和防火墙设备需升级支持相应协议，这对已大量部署的 NAT 和防火墙设备来说，是很困难的。STUN 方式采用客户机/服务器模式，无需 NAT 和防火墙的升级，但它无法穿越对称型 NAT。关于对称 NAT 型的穿越，业界既无标准协议可循也没有成熟的技术可供参考。这使得对对称型 NAT 的穿越成为当今的一个难点[17][18]。

本文依据 TURN 草案，设计并实现了一种能穿越对称型 NAT 的 NAT 穿越方案。它通过分配 TURN 服务器的地址和端口作为客户端对外的接受地址和端口，即局域网用户发出的报文都要经过 TURN 服务器进行中继转发，来实现对对称型 NAT 的穿越。由于 TURN 协议为客户机/服务器模式，因此也无需 NAT 的任何修改。

实现上，将 TURN 客户端移植在嵌入式 Linux 平台上，提供封装接口供 SIP 信令使用。

在此基础之上，本文提出一种将 STUN 方式与 TURN 方式相结合的 NAT 穿越方案。对于对称 NAT，使用 TURN 即通过中继方式进行穿越。应用模型通过分配服务器的地址和不同端口作为客户端对外的接受地址和端口，即私网用户发出的报文都要经过服务器进行中继转发。而对于非对称 NAT，则使用 STUN 进行穿越。

这种方式应用模型综合 STUN 和 TURN 两种不同的 NAT 穿越方式的优点，既解

决了 STUN 应用无法穿越对称型 NAT 的缺陷，又降低 TURN 服务器的负荷。

1.4 本文组织结构

本文第一章为绪论，介绍该课题的研究背景，国内外发展现状，阐述研究该课题的目的及意义，并提出该课题的创新点。

第二章介绍与该课题有关的技术。包括 NAT 及其分类，NAT 对 VoIP 的影响，并对此文所涉及的 STUN 和 TURN 技术做了较为详细的介绍。由于此方案实现在以 SIP 为信令的 VoIP 网关中，故对 SIP 信令也进行介绍。

第三章详细描述该系统的设计。包括总体方案、系统应用体系结构、功能组件划分，以及 TURN 服务器、客户端的模块设计。

第四章着重描述 TURN 服务器、客户端的实现、系统软件流程、系统消息流程。包括数据结构，函数定义，以及库的描述。

第五章描述了测试方案和测试拓扑的设计，将方案应用于基于 SIP 的 VoIP 网关中，对 NAT 穿越的基本功能和性能进行了测试，并给出了测试结果。

第六章对工作的成果进行了总结并探讨了下一步的研究方向。

2 相关技术简介

在 VoIP 网关中涉及到一些重要的相关技术如 SIP、SDP（Session Description Protocol）、RTP（Real-time Transport Protocol）[19]以及 NAT 穿越的一些关键技术如 STUN 协议、TURN 协议。其中 SIP、SDP、RTP 是和 NAT 穿越紧密相关的 VoIP 协议，STUN 协议和 TURN 协议是本文设计的 NAT 穿越方案的基础，本章对上述关键技术进行深入的分析。

2.1 NAT 及其分类

2.1.1 NAT 技术定义

NAT 是 IETF（Internet Engineering Task Force）提出的标准，用于解决 IPv4 地址匮乏和加强安全性。NAT 允许一个机构以一个地址出现在 Internet 上，它将局域网内的每个私有地址转换成一个公有 IP 地址，反之亦然。这样就可以合理地安排网络中的公有 IP 地址和私有 IP 地址的使用。它还可以应用到防火墙技术中，把个别 IP 地址隐藏起来不被外界发现，使外界无法直接访问内部网络设备。

2.1.2 NAT 技术原理

NAT 技术可以缓解 IP 地址紧缺问题[20][21]。其原理是： NAT 把内部网络中的各主机使用私有地址都翻译成一个共享的、可全局路由的公有 IP 地址，具体的做法是把 IP 包内的私有地址用 NAT 对外的合法 IP 地址来替换，从而节省了地址资源[22][23]。NAT 设备维护一个地址映射表，将私有 IP 地址映射成公网的 IP 地址，然后再转发出去[24]。NAT 的工作原理图如图 2.1 所示。

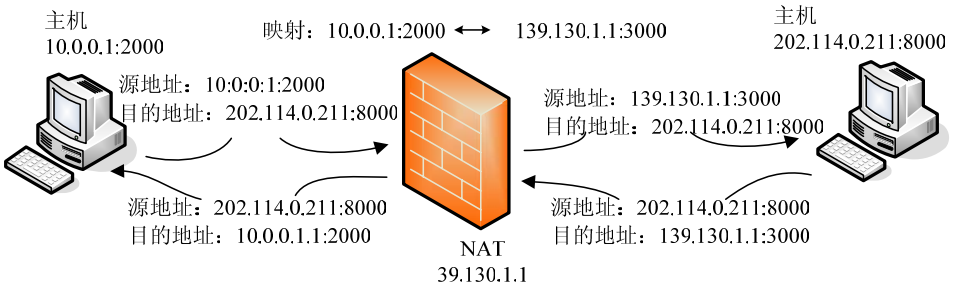

图 2.1 NAT 的工作原理图

2.1.3 NAT 的类型

按照处理端口映射的方式以及与 NAT 外主机通信的方式上是不同的，NAT 可以分为下面四种类型[25]，如图 2.2 所示：

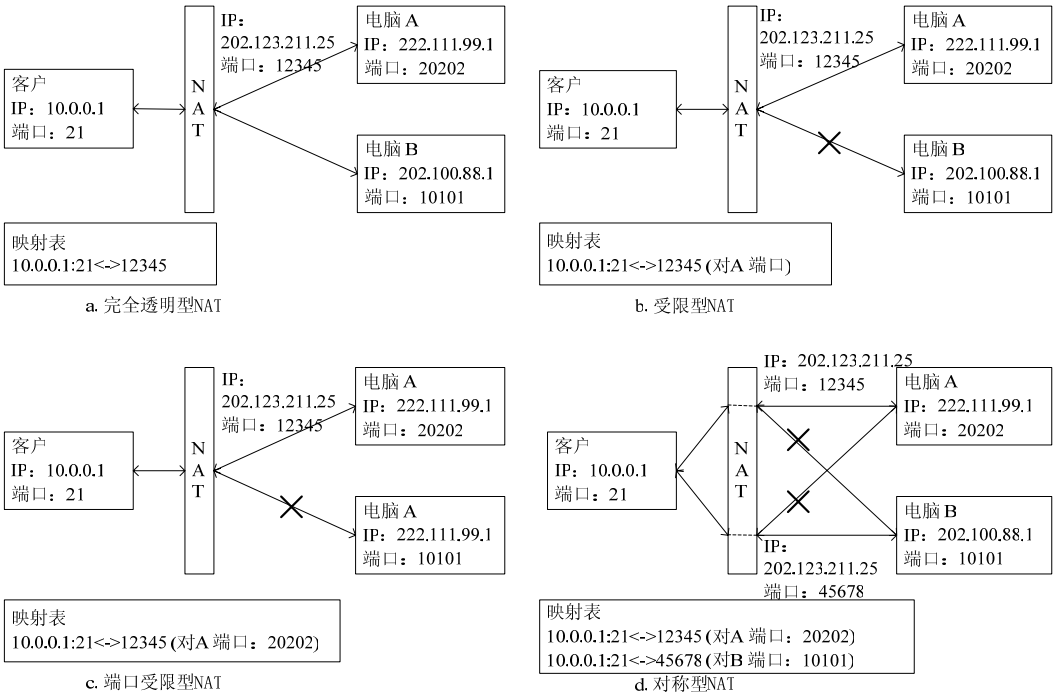

图 2.2 NAT 分类

完全透明型 NAT：把所有来自相同内部 IP 地址和端口的请求映射到相同的外部 IP 地址和端口。而且来自任何外部主机的数据包都能通过这个映射的外部地址和端口转发给内部主机。

受限型 NAT：把所有来自相同内部 IP 地址和端口号的请求映射到相同的外部 IP 地址和端口。只有当内部主机曾经给 IP 地址为 X 的外部主机发送过数据包时，来自 IP 地址为 X 的外部主机的数据包才能通过映射的外部地址和端口转发给内部主机。

端口受限型 NAT：端口限制型与地址限制型类似，只是多了端口限制。只有当内部主机曾经给 IP 地址为 X，端口号为 P 的外部主机发送过数据包，来自 IP 地址为 X，端口号为 P 的外部主机的数据包才能通过映射的外部地址和端口转发给内部主机。

对称型 NAT：把所有来自相同内部 IP 地址和端口号到特定目的 IP 地址和端口的请求映射到相同的外部 IP 地址和端口。但会根据目的地址的不同而映射到不同的地址和端口。只有当内部主机曾经给 IP 地址为 X，端口号为 P 的外部主机发送过数据包，来自 IP 地址为 X，端口号为 P 的外部主机的数据包才能通过映射的外部地址

和端口转发给内部主机。

2.2 NAT 对 VoIP 的影响

VoIP 中所使用的信令，如 SIP、H.323，在应用层数据包中包含后续 RTP 媒体流所需要地址和端口信息[26]。而 NAT 仅仅修改网络层和传输层的 IP 地址和端口，并不会修改应用层数据。这使得位于 NAT 内的信令，在经过 NAT 之后，应用层数据包中包含的地址和端口信息并未被修改成对外映射的公网地址，从而导致后续媒体流的连接失败。因此，在 VoIP 的应用中，必须解决 NAT 穿越问题。

2.3 STUN 技术

2.3.1 STUN 操作概述

应用程序即 STUN 客户端向 NAT 外的 STUN 服务器通过 UDP[27]发送 STUN 请求。STUN 服务器收到请求消息，产生响应，响应消息中包含请求消息的源端口，即 STUN 客户端在 NAT 上映射的外部端口。然后响应消息通过 NAT 发送给 STUN 客户端，STUN 客户端通过响应消息体中的内容得知其在 NAT 上对应的外部地址，并且将其填入以后信令协议的 UDP 负载中，告知对方自己的 RTP 接收地址和端口号为 NAT 外的地址和端口号[28][29]。由于通过 STUN 协议已在 NAT 上预先建立媒体流的 NAT 映射表，故媒体流可以顺利穿越 NAT。

STUN 协议最大的优点是无需现有 NAT 设备做任何改动[30]。目前，网络中已有大量的 NAT，而且这些 NAT 并不支持 VoIP 应用。如果采用 MIDCOM 或 ALG 方式，则需要替换现有的 NAT，实施起来难度较大，且 MIDCOM 方式无法实现对多级 NAT 的有效控制。如果采用 STUN 方式，不但无需改动 NAT，而且能够很好地适应多个 NAT 串联的网络环境。

但 STUN 的局限性在于需要应用程序支持 STUN 客户端的功能；同时 STUN 不支持 TCP[31]连接的穿越，因此不支持 H.323 协议；另外 STUN 方案不支持 NGN 业务对防火墙的穿越以及对称型 NAT（在安全性要求较高的企业网中，出口 NAT 通常就是采用这种类型）的穿越。

图 2.3 是 STUN 应用环境的典型配置图，STUN 客户端连接到私网 1 上，此网络通过 NAT1 连接到私网 2 上，私网 2 通过 NAT2 连接到公网上。

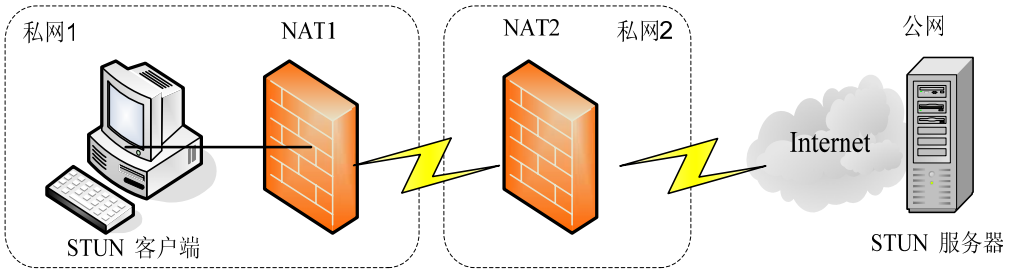

图 2.3 STUN 应用环境

STUN 是一个简单的客户机/服务器协议，客户机发送请求，服务器返回响应。STUN 定义了两种类型的请求：使用 TLS （Transport Layer Security）通过 TCP 发送的共享秘密请求和使用 UDP 发送的绑定请求。共享秘密请求希望服务器返回一个临时的用户名和密码，用于之后的消息认证和消息完整性检查；绑定请求用于得到 NAT 映射的公网地址和端口。

STUN 客户端一般被嵌入到需要得到公网 IP 地址和端口的应用程序中，获得的公网 IP 地址和端口是用来接收数据的。当应用程序启动时，嵌入到应用程序中的 STUN 客户端向服务器发送共享密钥请求获取临时用户名和密码，然后发送绑定请求。当绑定请求到达服务器的时候，它可能经过了一个或者多个位于 STUN 客户端 和 STUN 服务器之间的 NAT 设备。因此，服务器收到的请求的源地址是最靠近服务器的那个 NAT 映射的地址和端口，服务器把这个地址和端口拷贝到绑定响应中。当客户端收到绑定响应的时候，把响应中 MAPPED-ADDRESS 的值与发送绑定请求时的本地地址和端口做比较，如果不匹配，那么 STUN 客户端是位于一个或者多个 NAT 之后。

2.3.2 STUN 消息概述

STUN 消息是使用 TLU（type-length-value）编码的网络字节序格式。所有的 STUN 消息都以一个头部开始，后面跟紧一个负载，该负载是由一系列的属性组成的，属性的设置取决于消息类型。STUN 消息头部包含消息类型、事务 ID 和长度，事务 ID 用于关联请求和响应，长度表示负载的总长度，头部格式如图 2.4 所示。

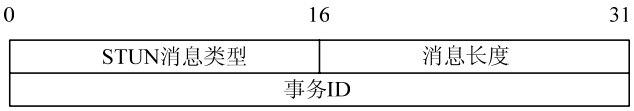

图 2.4 STUN 消息头部格式

消息类型及其对应的消息代码如下：

0x0001：Binding 请求

0x0101：Binding 绑定

0x0111：Binding 错误响应

0x0002：Shared Secret 请求

0x0102：Shared Secret 响应

0x0x12：Shared Secret 错误响应

在协议中定义了如下的一些属性（括号中的是响应代码）：

MAPPED-ADDRESS（0x0001）：它对应一对 IP 地址和端口，指明 NAT 映射的地址和端口，这个属性总是出现在绑定响应中，它指明了 STUN 服务器在绑定请求中看到的源地址。

RESPONSE-ADDRESS（0x0002）：包含一对 IP 地址和端口，出现在绑定请求中，指明绑定响应将要发送到哪里，这个属性是可选的，当不存在时，响应将发送到绑定请求的源地址和端口。

CHANGE-REQUES（0x0003）：包含两个标志，控制用于发送绑定响应的 IP 地址和端口，这两个标志是 "change IP"和"change port"，这个属性只能出现绑定请求中，两个标志在判断是处于地址受限 NAT 还是端口受限 NAT 之后时起作用，它们告诉 STUN 服务器从不同的源 IP 地址和端口发送绑定响应。

SOURCE-ADDRESS（0x0004）：仅仅出现在绑定响应中，指明响应发送的源地址和端口。

CHANGE-ADDRESS（0x0005）：出现在绑定响应中。如果客户端在请求中设置了"change IP"和"change port"标志时，服务器通过这个属性告知客户端 STUN Server 将从这个指定的地址和端口发送绑定响应。

USERNAME（0x0006）：出现在共享密钥响应中，共享密钥享用给客户端提供了一个临时的用户名和密码，这个属性也出现在绑定请求中，作为共享密钥的索引，共享密钥是用于对绑定请求进行信息完整性保护的。

PASSWORD（0x0007）：临时密钥，仅出现在共享密钥响应消息中。

MESSAGE-INTEGRITY（0x0008）：包含了对绑定请求和绑定响应进行完整性检查的信息。

ERROR-CODE（0x0009）：指明了具体的错误代码。出现在绑定错误响应和共享密钥错误响应中。

UNKNOWN-ATTRIBUTES（0x000a）：指明请求中未知的属性。出现在绑定错误响应或者共享错误响应中，

REFLECTED-FROM（0x000b）：出现在绑定响应中，指出绑定请求发送者的 IP

地址和端口，用于跟踪以阻止一定的 DoS 攻击。

2.3.3 STUN 协议行为

STUN 包含 STUN 客户端和 STUN 服务器两个实体，所以 STUN 的行为也分为 STUN 客户端行为和 STUN 服务器行为。

1 客户端的行为

客户端的任务主要是发现 STUN 服务器，获取共享密钥，生成绑定请求，处理绑定响应。

（1）发现 STUN 服务器

通常 STUN 客户端被配置在能够找到 STUN 服务器的域中，这个域通常就是应用程序所需服务的提供者所在的域，或者也可以通过其他方法得到 STUN 服务器的地址，比如可以预先告诉用户 STUN 服务器的地址。

（2）获取共享密钥

为了提供完整性检验，STUN 在客户端和服务器之间采用了共享密钥机制，协议中采用的是基于 TLS 的机制，TLS 是承载在 TCP 之上的。客户端首先确定 STUN 服务器的地址和端口。STUN 客户端向上述地址和端口建立一个 TCP 连接，同时立刻开始 TLS 握手，一旦连接建立成功，STUN 客户端发送共享密钥请求，这个请求没有任何属性，只有头部。STUN 服务器产生响应，这个响应可能是共享密钥响应或者共享密钥错误响应。如果客户端收到共享密钥响应，在共享密钥响应中会包含一个临时的用户名和密码，分别对应 USERNAME 和 PASSWORD 这两个属性。客户端可以在这个连接上产生多个共享密钥请求，而且可以在收到上次的共享密钥响应之前做这些动作。一旦完成了获取用户名和密码的过程，客户端应该关闭连接。

（3）生成绑定请求

绑定请求消息是用来获取 NAT 映射的公网地址的。STUN 客户端遵循一定的语法规则构造绑定请求，任何两个不同的请求以及原地址和目的地址分别不同的请求必须携带不同的事务 ID，事务 ID 都随机分布在 0 到 255 的范围内。规定一个最大的范围是有必要的，因为事务是随机的，可以帮助阻止以前收到的绑定响应的重放。一旦消息构造成功，客户端发送请求，可靠性通过客户端的重传机制完成。

（4）处理绑定响应

响应有可能是绑定响应或者是绑定错误响应。绑定错误响应总是在发送请求的源地址和端口上收到的。绑定响应将在请求中的 RESPONSE-ADDRESS 属性所指示的地址和端口上收到，如果 RESPONSE-ADDRESS 不存在，绑定响应将在在发送请

求的源地址和端口上收到的。当收到的是绑定错误响应时，客户端应该根据响应中ERROR-CODE 属性的值来做出相应的动作。如果客户端收绑定响应，客户端应该检查响应的 MESSAGE-INTEGRITY 属性，如果不存在，则丢弃这个响应。如果存在，则对消息进行 HMAC 认证，如果认证失败则丢弃这个响应，并且警告用户可能存在的攻击，如果认证成功，那么客户端可以使用 MAPPED-ADDRESS 的值。

2 服务器端的行为

服务器端的行为取决于服务器收到的请求是绑定请求还是共享密钥请求，对于不同的请求服务器将做出相应的处理。

（1）处理绑定请求

STUN 服务器必须准备在以下四个地址和端口组合接收绑定请求：(A1，P1)，(A2，P1)，(A1，P2)，(A2，P2)。(A1，P1)代表的是 STUN 服务器的主地址和端口，一般地 P1 为 3478，是默认的 STUN 端口。A2，P2 是任意的，它们是 STUN 服务器通过 CHANGE-ADDRESS 属性推荐的。首先服务器必须检查帮定请求的信息完整性属性，之后检测请求中是否有属性类型小于或等于 0x7ffff 的属性，这些属性 STUN 服务器是无法理解的。如果绑定请求是正确的话，服务器必须生成一个合法的绑定响应。服务器不会重传这些响应，可靠性是通过客户端周期性地重发请求来达到。

（2）处理共享密钥请求

共享密钥请求一定是在 TLS 连接上收到的。当服务器收到来自客户端的建立 TLS 连接的请求时，它就必须处理 TLS 连接请求。如果服务器收到一个共享密钥请求，首先确认该请求是通过 TLS 连接收到的。如果共享密钥请求是合法的，服务器必须创建一个共享密钥响应，共享密钥响应是通过接收共享密钥请求的 TLS 连接发送的。服务器应该保持连接一直打开，由客户端来关闭它。

2.4 TURN 技术

TURN 方式解决 NAT 问题的思路与 STUN 相似，也是基于局域网接入用户通过某种机制预先得到其私有地址对应在公网的地址，不同的是 STUN 方式得到的地址为出口 NAT 上的地址，TURN 方式得到地址为 TURN 服务器上的地址，然后在报文负载中所描述的地址信息直接填写该公网地址的方式。TURN 适合处于对称 NAT 或防火墙之后的实体接收来自连接另一端的对等实体的数据。

TURN 通过分配 TURN 服务器的地址和端口作为客户端对外的接受地址和端口，即局域网用户发出的报文都要经过 TURN 服务器进行中继转发，这种方式应用

模型除了具有 STUN 方式的优点外，还解决了 STUN 应用无法穿越对称 NAT 以及类似的防火墙设备的缺陷，即无论出口为哪种类型的 NAT/FW，都可以实现 NAT 的穿越。

同时 TURN 支持基于 TCP 的应用，如 H.323 协议。此外 TURN 服务器控制分配地址和端口，能分配 RTP/RTCP 地址对（RTCP 端口号为 RTP 端口号加 1）作为本地客户端的接受地址，避免了 STUN 应用模型下出口 NAT 对 RTP/RTCP 地址端口号的任意分配，使得客户端无法收到对方发过来的 RTCP 报文（对端发送 RTCP 报文时，目的端口号缺省按 RTP 端口号加 1 发送）。TURN 的局限性在于需要终端支持 TURN 客户端，这一点同 STUN 一样对网络终端有要求。此外，所有报文都必须经过 TURN 服务器转发，增大了包的延迟和丢包的可能性，同时对 TURN 服务器的要求非常高。

2.4.1 TURN 操作概述

如图 2.5 所示是 TURN 的典型配置图。TURN 客户端连接到内网 1 上，此网络通过 NAT1 连接到内网 2 上，内网 2 通过 NAT2 连接到公网上，在公网上的是 TURN 服务器。

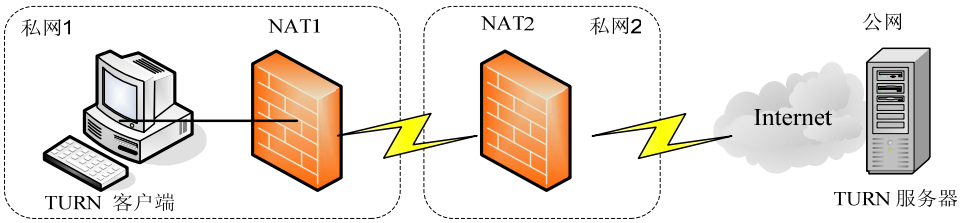

图 2.5 TURN 应用环境

TURN 是一个简单的客户机/服务器结构协议。为了方便两者的共同实现，TURN 和 STUN 在语法和基本操作方面是相同的。TURN 既可以承载于 UDP 之上，也可承载于 TCP 之上。

TURN 定义了 Allocate 请求消息，该请求可以让服务器给客户端分配公网 IP 地址和端口。TURN 客户端首先得到 TURN 服务器的地址，然后客户端发送 TURN 分配请求到服务器，TURN 协议为请求和响应提供了一种基于共享密钥的相互认证和消息完整性检查的机制，假设请求通过了认证并且没有被篡改，服务器将给客户端分配一个传输地址，并且把这个地址通过 Allocate 响应返回给客户端。

TURN 服务器转发数据要遵循一定的规则。只有当客户端通过 TURN 服务器先给对端实体发送数据之后，TURN 服务器才会将数据包从 PA（Public Address）转发

到 SA（Source Address）。为此，客户端发送 Send 请求，该请求中包含一个数据包和一对目的 IP 地址和端口，当服务器收到该请求时，就把其中的数据包转发到 Send 请求中指定的目的地址，并把该地址添加到服务器的转发权限表中，以后只要是来自这个地址和端口的数据包都会被转发。

通过 Send 请求激活的 UDP 或 TCP 通道收到的数据包会被 TURN 服务器封装成 Data Indication 消息后转发给客户端，所以使用 Send 请求和 Data Indication 消息的效率很低，因此当客户端决定和哪个外部客户端进行通信时，它可以发布一个设置 Set Active Destination 请求，该请求告诉服务器以后非 TURN 格式的数据都不需要封装成 Data Indiction 消息就可以被转发。比如，如果客户端向 TURN 服务器发送普通数据包，它将会被服务器转发到这个预先设置的活跃目的地址；相似地，从外部客户端收到的包也不用封装成 Data Indication 消息就会被转发到客户端。为此 TURN 服务器必须在一个内部五元组和一个或多个外部五元组之间维持一个绑定，如图 2.7 所示。内部五元组代表了 TURN 服务器和 TURN 客户端之间的连接，对于 UDP 来说，它是 TURN 客户端发送分配请求时的源地址和端口与目的地址和目的端口之间的组合。外部本地传输地址是 TURN 服务器分配给 TURN 客户端的公网 IP 地址和端口，外部五元组是外部本地传输地址和外部客户端的 IP 地址和端口之间的组合。一开始并不存在外部五元组，因为 TURN 客户端没有和任何主机通信，当在分配的传输地址上收到数据包或者发送数据包的时候外部五元组就被创建了。

2.4.2 TURN 消息概述

TURN 消息和 STUN 消息在语法上是一样的，也是由消息头和消息体组成，消息体是消息属性的组合。TURN 消息头的格式如图 2.6 所示。

图 2.6 TURN 消息头部格式

TURN 定义了一些新的消息，消息类型和对应的代码如下：

0x0003：Allocate 请求

0x0103：Allocate 响应

0x0113：Allocate 错误响应

0x0004：Send 请求

0x0104：Send 响应

0x0114：Send 错误响应

0x0115：Data Indication 消息

0x0006：Set Active Destination 请求

0x0106：Set Active Destination 响应

0x0116：Set Active Destination 错误响应

TURN 也使用 STUN 中定义的共享密钥请求，共享密钥响应和共享密钥错误响应。TURN 使用 STUN 中的一些属性，也定义了如下新的属性：

LIFETIME（0x000d）：服务器通过该属性告知客户端分配绑定的维持时间。

ALTERNATE-SERVER（0x000e）：另外可选的 TURN 服务器地址。

MAGIC-COOKIE（0x000f）：客户端用来区分普通数据和 TURN 消息。

BANDWIDTH（0x0010）：指明客户端期望使用的连接带宽。

DESTINATION-ADDRESS（0x0011）：指明数据包将会被发送到何处，出现在 Send 请求中。

REMOTE-ADDRESS（0x0012）：告知客户端数据包来自哪里的，出现在 DATA Indication 中。

DATA（0x0013）：包含了在 DATA Indication 中的数据内容。

NONCE（0x0014）：出现在共享密钥请求和共享密钥错误响应中，用于消息认证。

REALM（0x0015）：出现在共享密钥请求和共享密钥错误响应中，用于消息认证。

2.4.3 TURN 协议行为

TURN 协议包含客户端和服务器两个实体，所以 STUN 的行为也分为 TURN 客户端行为和 TURN 服务器行为。

TURN 客户端行为：客户端的行为分为几个独立的步骤，它的任务主要包括服务器发现过程，获取临时密码，分配绑定，处理分配响应，更新绑定，发送封装数据，接收 Data Indication，设置主动目的，地撕裂绑定，接收和发送未封装数据。

TURN 服务器行为：服务器的行为取决于请求是共享密钥请求还是分配请求或者是设置活跃目的地请求。它的主要任务包括共享密钥请求，分配请求，Send 请求，在分配的传输地址上接收数据包，接收主动目的请求，接收来 TURN 客户端的数据，生存时间到期检查。

2.5 SIP 技术

2.5.1 SIP 基本概念和功能

SIP 是一个应用层的控制协议，可以用来建立、修改、和终止多媒体会话（或者会议）[32] [33]。SIP 也可以邀请参与者参加已经存在的会话，比如多方会议。媒体可以在一个已经存在的会话中方便的增加（或者删除）[34]。SIP 显示的支持名字映射和重定向服务，这个用于支持个人移动业务－用户可以使用一个唯一的外部标志而不用关系他们的实际网络地点。SIP 在建立和维持终止多媒体会话协议上，支持 5 个方面

(1) 用户定位：确定参与通信的终端系统的位置；
(2) 用户能力：确定通信所使用的媒体类型及媒体参数；
(3) 用户可用性：确定被叫方是否愿意加入通信；
(4) 呼叫建立：在主、被叫之间建立约定的、支持特定媒体流传输的连接；
(5) 呼叫处理：包括呼叫修改，呼叫转移和呼叫终止等处理。

2.5.2 SIP 网络体系结构

SIP 是基于客户机/服务器结构的协议，SIP 网络结构的组成包括：SIP 用户代理（User Agent，UA），SIP 代理服务器（SIP Proxy Server）、SIP 重定向服务器（SIP Redirect Server）、SIP 注册服务器（SIP Registrar）和定位服务器[35]。SIP 用户代理是终端组件，包含两个功能实体：用户代理客户端（User Agent Client，UAC）和用户代理服务器（User Agent Server，UAS）[36]，UAC 负责发起 SIP 呼叫请求，UAS 负责接受呼叫并做出响应，二者组成用户代理存在于用户终端中。SIP 代理服务器代表客户机发起请求，对收到的请求消息进行处理后转发传递给其他服务器。SIP 重定向服务器把 SIP 请求中的原地址映射成零个或多个新地址，返回给客户机。SIP 注册服务器接收客户机的注册请求，完成用户地址的注册。定位服务器的提供位置查询服务 [37]，主要是由代理服务器或重定向服务器用来查询被叫的可能的地址信息。定位服务器本身不属于 SIP 服务器。SIP 的网络结构图如图 2.7 所示。

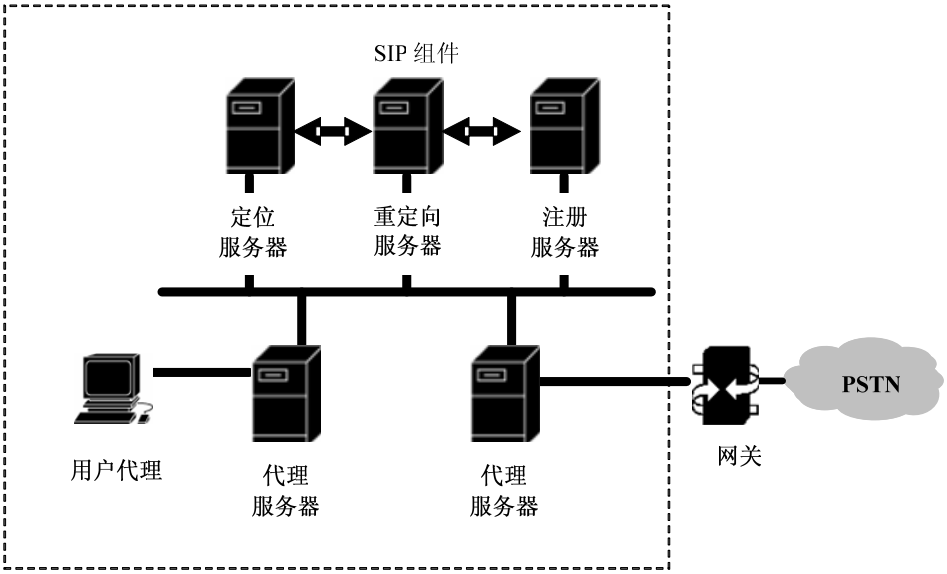

图 2.7 SIP 网络结构图

2.5.3 SIP 消息介绍

SIP 是一个基于文本的协议，SIP 消息分为两大类：请求消息和响应消息。

SIP 协议定义了六种请求消息：

(1) REGISTER ：用于向注册服务器注册客户机的相关信息。

(2) INVITE ：初始化一个呼叫。

(3) ACK ：确认客户机已经接收到对 INVITE 的最终响应。

(4) CANCEL ：取消正在进行中的请求。

(5) BYE ：终止呼叫，结束会话。

(6) OPTIONS ：询问远端支持的特性。

SIP 协议定义了六种响应消息：

(1) 1XX：临时响应消息，表示请求已经被接收，正在处理中。

(2) 2XX：成功响应，表示请求已经成功的接收到，理解并接受。

(3) 3XX：重定向，表示为了处理请求需要进一步操作。

(4) 4XX：用户错误，表示请求包含错误文法，服务器不能满足。

(5) 5XX：服务器错误，表示服务器无法响应显然有效的请求。

(6) 6XX：全局错误，表示任何服务器都无法响应该请求。

SIP 消息由以下几部分组成：起始行，消息头，CRLF，消息体。其格式为：

```
Generic-message = Start-line
*message-header
CRLF
 [message-body]
```

请求消息的起始行被称为请求行，响应消息的起始行被称为状态行。请求行包括方法类型、请求 URI 和协议版本，请求行必须以 CRLF 字符表示终结。状态行包括协议版本、状态码、与状态码相关的文本描述。状态行也以 CRLF 表示终结。

消息头用来描述消息的属性，其格式为<名字> ：<值>。消息头又分为通用消息头、请求消息头、响应消息头、实体消息头。每个头域都以 CRLF 作为结束标志。SIP 消息头域的详细介绍请参考 RFC3261，下面是 SIP 消息中一些重要的头域：

From：指示请求的发起者。

To：指示请求消息的接收者，语法形式与 From 完全相同。

Call-ID：唯一标识一个特定的请求或者一个特定客户的所有登记。

Request-URL：用来指示请求消息的寻址，它的值可以被代理服务器修改。

CSeq：由请求方法和十进制数字组成。相同 Call-ID 发出的请求必须使用单调增长的数字。ACK 和 CANCEL 请求使用和初始 INVITE 请求相同的 Cseq 值。

Contact：用于给后续的消息提供联系地址。

Via：指示请求迄今为止所走的路径，要求响应消息按同样的路径返回。

Content-Type：指示发送给接收者的消息体的媒体类型。

Content-Encoding：它的值指示适用于实体消息体的其他的内容编码。

Content-Length：指示消息体的长度，形式上以八个比特为一个字节。

消息体是对消息所要建立的会话的描述，比如建立一个多媒体会话的消息体中包含音频/视频编码类、取样频率等信息的描述，典型的消息体为 SDP 格式。

2.5.4 SIP 呼叫流程

SIP 协议建立通信的流程主要包括：注册、呼叫连接建立、媒体流通道建立、呼叫更改或处理、呼叫终止。图 2.8 是典型的 SIP 通信流程图。首先主叫向被叫发出 INVITE 请求，被叫接收到请求后作出应答，当主叫收到应答后发送 ACK 请求，主叫或被叫在呼叫建立后发起后续请求[38] [39]。

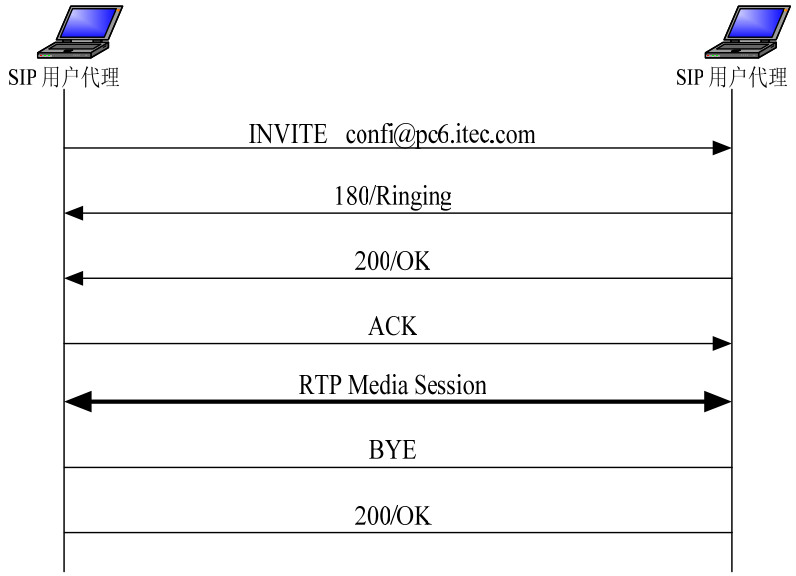

图 2.8 SIP 用户代理直接呼叫流程

2.5.5 会话描述协议

SDP 是会话描述协议[40]，目的是传送多媒体会话中的媒体信息[41]。目前 SIP 主要采用 SDP 进行媒体协商，SDP 包含的媒体信息主要包括：

(1) 媒体类型，例如视频和音频。

(2) 传输协议，例如 RTP/UDP/IP 和 H.320。

(3) 媒体格式，例如 H.261 视频和 MPEG 视频。

(4) 多播地址和媒体传输端口（IP 多播会话）。

(5) 用于联系地址的媒体和传输端口的远端地址（IP 单播会话）[42]。

SDP 描述由许多文本行组成，文本行的格式为<类型>=<值>，<类型>是一个字母，<值>是结构化的文本串，其格式依<类型>而定。

下面是 SDP 的一个例子：

```
----------SDP Message ----------
v=0
o=xjq 20513597 3520648192 IN IP4 192.168.88.103
s=Session SDP
c=IN IP4 192.168.88.103
m=audio 8009 RTP/AVP 0
```

2.6 实时传输协议

RTP 协议是为支持实时多媒体数据通信而设计的传输层协议，它由 IETF 作为

RFC1889 发布。它位于 TCP 和 UDP 协议层之上，通常是利用低层的 UDP 协议对实时音/视频数据进行组播或单播，从而实现多点或单点音/视频数据的传输。它本身并不提供任何传输可靠性的保证和流量的拥塞控制机制，需要 RTCP 实时监控数据传输和服务质量。在实际应用中， RTP/RTCP/UDP 多用于音/视频媒体数据的传输， 而 TCP 则主要用于传输、交互控制信令等。

2.7 本章小结

本章介绍与本文有关的 NAT 穿越相关技术，以及 NAT 对 VoIP 的影响。了解 NAT 技术的定义、NAT 工作原理、NAT 的类型对于整个 NAT 穿越方案的设计至关重要，本章给出了这些概念的详细介绍。STUN 协议和 TURN 协议是涉及整个 NAT 穿越方案的基础，因此本章对 STUN 协议和 TURN 协议作了较为细致的介绍和分析。主要包括相应的基本操作、消息格式以及客户端、服务器的行为。由于本 NAT 穿越方案实现在 VoIP 嵌入式平台上，后者涉及了众多的相关技术，其中 SIP 协议、SDP 协议以及 RTP 协议与 NAT 穿越方案的设计和实现紧密相关，所以本章也详细介绍了 VoIP 网关中的信令控制协议 SIP，包括 SIP 协议的基本概念和功能、SIP 基本网络体系结构、SIP 消息的格式以及 SIP 典型通信流程。然后对多媒体会话描述协议 SDP 做了简单的介绍，主要包括了 SDP 的功能和消息格式。

3 系统设计

上一章分析的各种方案各有优缺点，譬如ALG方式和MIDCOM方式都需要NAT设备支持。我们的 VoIP 网关是放在 NAT 之后的设备，而这两种方案是不能穿越别人的 NAT，所以首先被排除。TURN 方式是不需要 NAT 设备支持，但需要在公网上架设 TURN 服务器，而且转发所有数据包会增加服务器负担，成为性能瓶颈。而 STUN 方式只需要在公网上架设一个服务器来处理 STUN 客户端请求，就可以保证穿越所有非对称的 NAT，而不会增加 NAT 和服务器负担，但它却无法穿越对称 NAT。结合考虑多种 NAT 穿越方式的优缺点，本章设计了一种基于 TURN 协议的 NAT 穿越方案，主要包括总体方案设计，软件功能划分和 TURN 模块设计。下面详细描述各个部分的实现。本文在总体方案设计、系统软件流程以及系统消息流程中，均将 STUN 与 TURN 结合考虑。而在服务器和客户端模块的实现中，均只涉及 TURN 部分。该部分为本文的核心。

3.1 总体方案

本文参考 STUN 草案以及 TURN 草案，选择 STUN 与 TURN 结合的方式穿越所有类型的 NAT。使用 STUN 完成 NAT 是否存在以及 NAT 类型的检测，并获知 STUN 服务器分配的绑定地址。如果 NAT 设备为非对称 NAT，则使用 STUN 进行穿越；如果 NAT 设备为对称 NAT，则使用 TURN 进行穿越。这样既结合了 STUN 不会增加 NAT 和服务器负担的优势，又使用 TURN 弥补了无法穿越对称型 NAT 的缺陷。

3.2 系统应用体系结构

采用 STUN 和 TURN 结合的方式的系统应用体系结构如图 3.1 所示。STUN 协议和 TURN 协议均是简单的客户/服务器协议。在这里 STUN 服务器和 TURN 服务器被放置在公网，它们可以放在同一台主机中，只要端口不冲突就可以。在本系统中，为了减轻运行 TURN 服务器的主机中继的负担，将 STUN 服务器和 TURN 服务器分别运行在两台主机上。而 NAT 内的客户端上，运行有 STUN 客户端和 TURN 客户端。

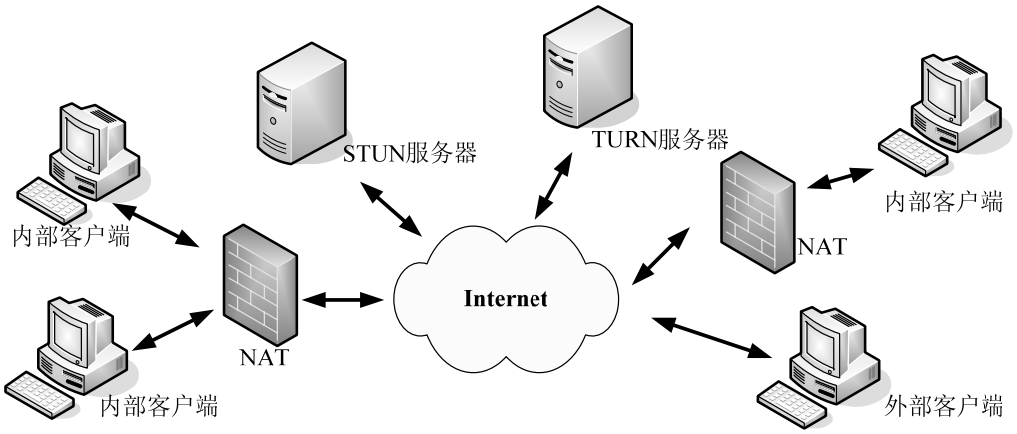

图 3.1 系统应用体系结构图

图中各部分功能表述如下：

NAT：位于 Internet 上的 NAT 设备，用于使用另一个地址来替换 IP 头部中源地址或目的地址。它最常用于在一个专用的、已编址本地址网络和使用公用的、可寻址的 Internet 之间进行出站连接的映射。

STUN 服务器：位于 Internet 上，提供所有非对称 NAT 的穿越。它的功能是负责接收来自 STUN 客户端的绑定请求和共享秘密请求。

TURN 服务器：位于 Internet，提供对称 NAT 的穿越。它的功能是负责接收来自 TURN 客户端的分配请求（Allocate Requests）、发送请求(Send Requests)、设置活跃地址请求(Set Active Destination Request)。

内部客户端：位于 NAT 设备之后的用户代理客户端。它上面运行有 STUN 和 TURN 的客户端程序，用于 NAT 的穿越。

外部客户端： 位于 Internet 上的用户代理客户端。

3.3 TURN 功能组件划分

TURN 是基于客户机/服务器结构的协议，它包括以下两个功能组件。

TURN 客户端：生成 TURN 请求消息，可以运行在一个端系统中，比如用户的 PC，也可以运行在网络元素中，比如会议服务器。

TURN 服务器：接收 STUN 请求，发送 TURN 响应，TURN 服务器位于公网上。

3.4 TURN 服务器模块设计

TURN 服务器包含消息解析器、消息生成器，将一次会话中用户名、密码、源地址、绑定端口以及生命期联系在一起的绑定模块，还有数据转发模块，如图 3.2 所示。

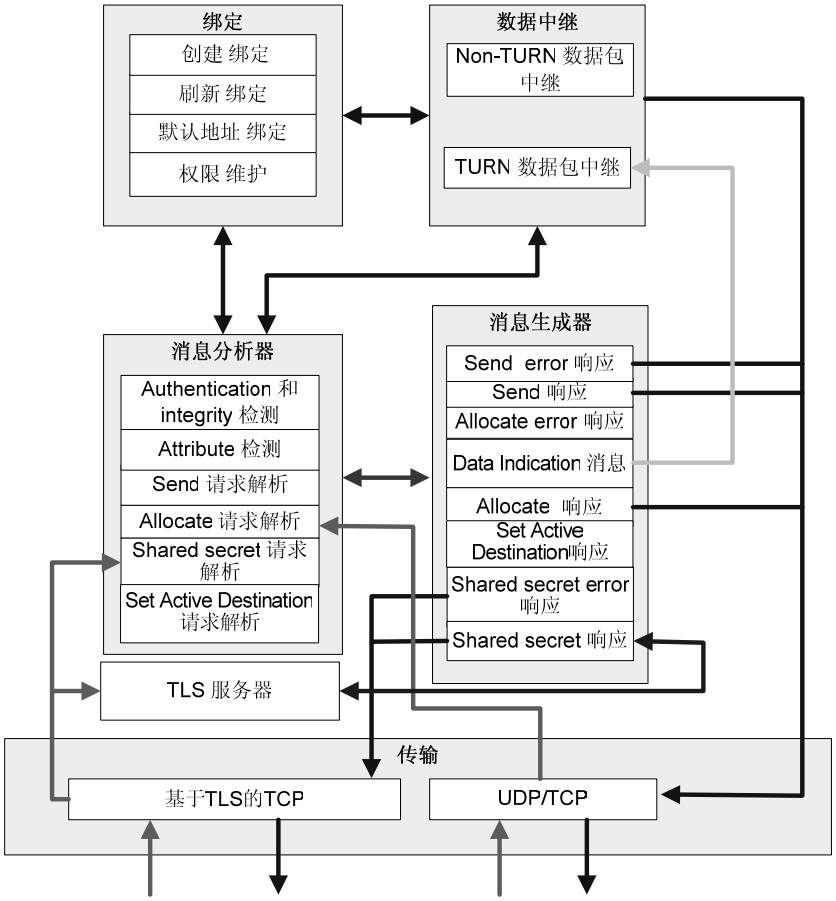

图 3.2 TURN 服务器模块设计

3.4.1 请求消息分析模块

服务器的行为是被动的，它根据收到的消息来触发具体的行为。服务器启动之后，监听默认端口，看是否有数据到来。当服务器收到请求消息时就交给消息分析器处理。消息分析器会先检测消息的身份和完整性，再检测其属性。TURN 消息和 STUN 消息在语法上是一样的。TURN 定义了几个新的消息，其中请求消息包括：Shared Secret 请求、Allocate 请求、Send 请求和 Set Active Destination 请求。

3.4.2 响应消息生成模块

消息分析器根据消息属性的不同进行相应的解析，并将消息的处理结果告知消息生成模块。消息生成模块根据处理结果创建不同的响应，然后发回给客户端。TURN 响应消息的类型有：Shared Secret 响应、Shared Secret 错误响应、Allocate 响应、Allocate 错误响应、Send 响应、Send 错误响应、Set Active Destination 响应、Set Active Destination 错误响应和 Data Indication。

3.4.3 绑定模块

TURN 服务器在一个内部五元组和一个或多个外部五元组之间维持一个绑定，如图 3.3 所示。

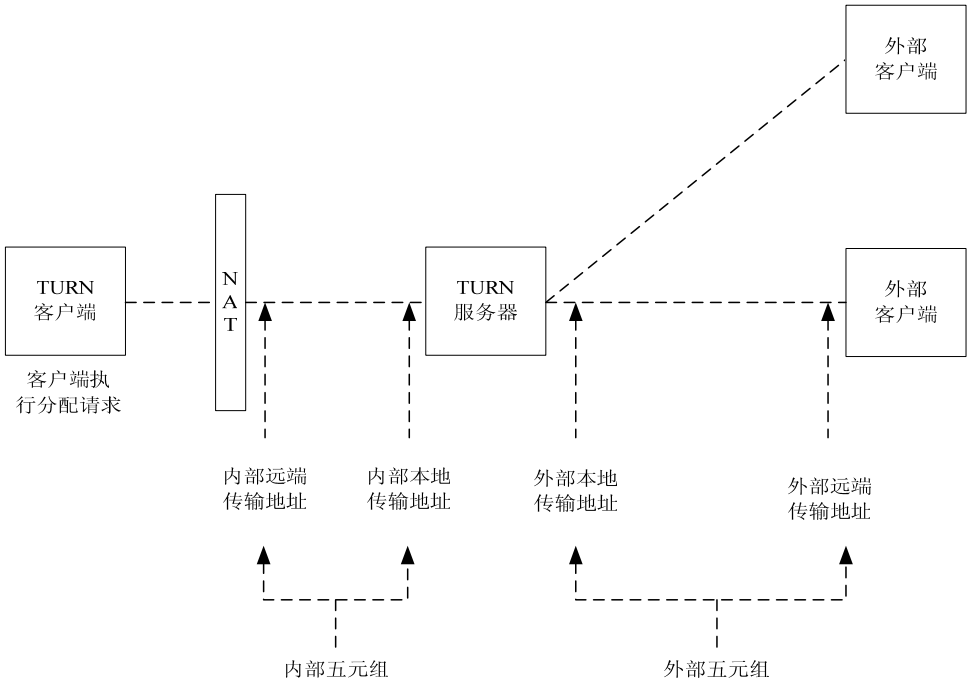

图 3.3 TURN 服务器五元组组合示意图

内部五元组代表了 TURN 服务器和 TURN 客户端之间的连接，它是 TURN 客户端发送分配请求时的源地址和端口与目的地址和端口之间的组合。外部本地传输地址是服务器分配给客户端的地址和端口。外部五元组是外部本地传输地址和 TURN 客户端通过 TURN 服务器与之通信的外部客户端的 IP 地址之间的组合。一开始，并不存在外部五元组，因为 TURN 客户端还没有和任何其它的主机通信。当在分配的传输地址收到数据包或者发送数据包的时候，外部五元组就被创建了。在设定的时

间周期内，如果没有收到数据包，TURN 服务器可以中止这个连接。这个时间周期是通过对应分配请求的分配响应发送给客户端的。

3.4.4 数据中继模块

仅当内部客户用户通过 TURN 服务器向公网用户发送过一个数据包，TURN 服务器才会将该公网用户的数据包转发给内部客户端。否则，TURN 服务器是不会将任何数据报从公网用户中转到内部用户的。为此，客户端将发送一个 Send 命令，这个命令包含了一个数据包和一个目的 IP 地址和端口，TURN 服务器一旦收到这个命令，就会把其中的数据包转发到 Send 命令中指定的目的地址，并把这个地址添加到 TURN 服务器“允许”列表中，以后只要是从那个地址和端口进的数据包都回被允许通过。

3.5 TURN 客户端模块设计

TURN 客户端包含消息解析器、消息生成器以及发现模块。TURN 客户端的消息生成器负责 Allocate 请求、Send 请求的构造，消息解析器负责解析来自于 TURN 服务器的应答并做出相应的处理。图 3.4 为 TURN 客户端模块图。

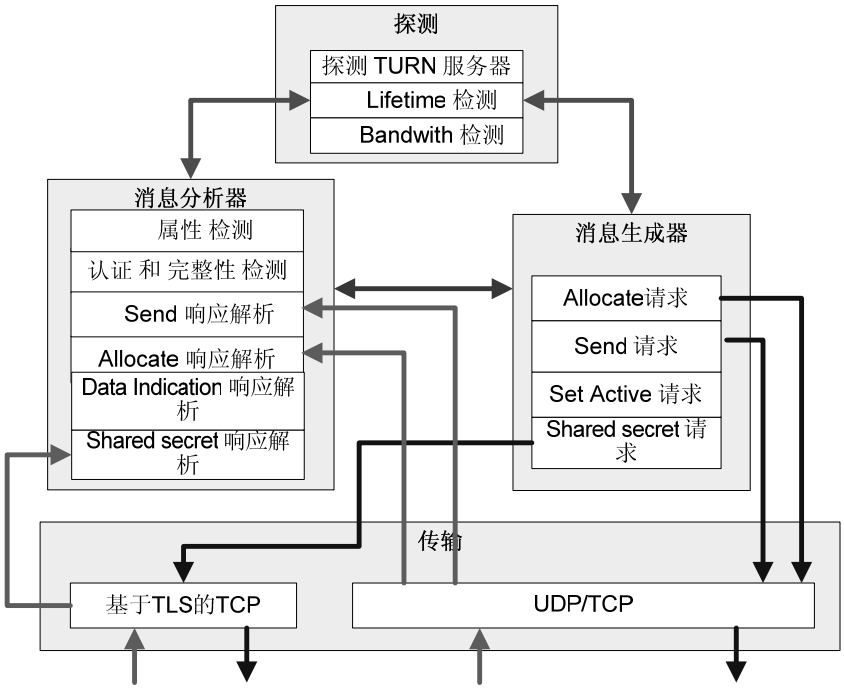

图 3.4 TURN 客户端模块

3.5.1 探测模块

TURN 客户端首先需要探测 TURN 服务器的地址，这可以通过预先配置方式获得。一旦发现了服务器，客户端就发送一个 TURN Allocate 请求到 TURN 服务器，TURN 为请求和响应提供了一种基于共享密钥的相互认证和消息完整性检测的机制。假定请求通过了认证，并且没有被篡改，服务器将给客户端分配一个传输地址。并且把这个传输地址通过分配请求对应的分配响应返回给客户端。通常，这个传输地址时服务器上众多接口之一。

3.5.2 请求消息生成模块

该模块负责请求消息的生成。请求消息产生模块会根据需要产生 Shared Secret 请求、Allocate 请求或 Send 请求。Shared Secret 请求是用来获取临时用户名和密码的，该临时用户名和密码用于随后的请求和响应的认证。由于本文设计的 TURN 客户端将运行在基于 SIP 的嵌入式平台上，故可以将身份认证交由该嵌入式平台上的其他应用程序如 SIP 完成。即，可以认为已通过 SIP 认证的用户也拥有使用 TURN 的权限。为降低设计和实现的难度，本文依照此省略 Shared Secret 请求。

Allocate 请求用来获取 IP 地址和端口，客户端可以使用这个 IP 地址和端口来接受来自外部网络任何主机的数据包，甚至当客户端位于对称 NAT 之后。为了在服务器中建立许可权限和向外部客户端发送数据，客户端完成 Allocate 事务之后就会发送 Send 请求。通过 Send 请求打开的 UDP 或 TCP 连接收到的数据包会被服务器封装成 Data Indication 消息后转发给客户端。使用 Send 请求和 Data Indication 消息的效率很低。因此，一旦客户端决定希望和哪个外部客户端进行通信，它可以发送一个 Set Active Destination 请求。这个请求告诉服务器以后未封装的数据都会被转发到这个发布的目的地址。比如，如果客户端向 TURN 服务器发送非 TURN 数据包，它将会被服务器转发到这个发布的目的地址去。相似地，从外部客户端收到的包也会不用封装成 Data Indication 消息就会被转发到客户端。

3.5.3 响应消息分析模块

响应消息处理模块负责对服务器返回的响应进行处理。对应 Allocate 请求的响应可能是 Allocate 响应或 Allocate 错误响应。收到响应时，需要对响应消息应进行认证和属性检查。客户端可以提取 Allocate 响应中的地址映射信息。对于 Allocate 错误响应，客户端根据响应中的错误代码做出相应的处理。同样，对应 Send 请求的响应

可能是 Send 响应或 Send 错误响应

3.6 本章小结

NAT 穿越已经应用的比较广泛，但对称型 NAT 的穿越却一直是业界的难点。本章着重阐述了 TURN 服务器和 TURN 客户端的具体模块设计，解决对对称型 NAT 的穿越。另外，本章提出了将 STUN 与 TURN 结合穿越 NAT 的观点，既保证了对所有类型 NAT 的穿越，又降低了 TURN 服务器的负荷，使整个系统有更好的稳定性和可实用性。

4 系统实现

第三章设计了基于 TURN 协议的穿越对称 NAT 方案，TURN 协议作为设计方案的基础，TURN 模块在整个方案的设计中处于非常重要的地位，为此本章首先根据 TURN 客户端、TURN 服务器的设计方案详细的描述了 TURN 客户端、TURN 服务器的实现过程。在实现 TURN 模块的基础上，详细描述了本文设计的基于 TURN 协议 SIP 穿越对称 NAT 的整体消息流程。

4.1 系统软件流程

图 4.1 显示的是系统软件流程图。

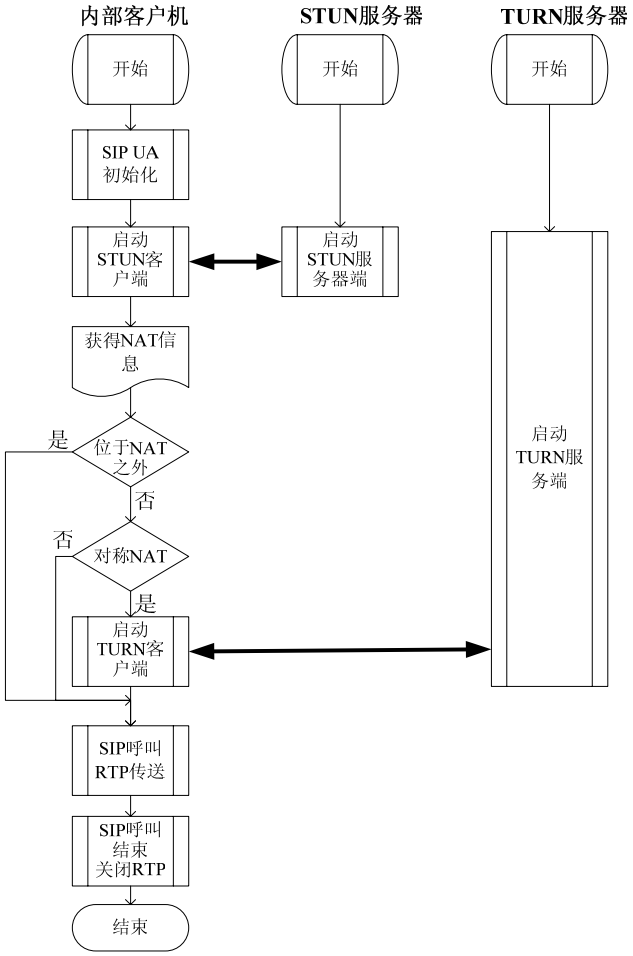

图 4.1 系统软件流程图

内部客户端在 SIP 用户代理初始化之后，启动 STUN 客户端，STUN 客户端通过与 STUN 服务器通信，检测它和 STUN 服务器之间是否存在 NAT，并得到 NAT 的类型、NAT 映射的公网地址以及绑定的生存周期等信息。如果存在 NAT 而且是位于非对称类型 NAT 之后，则接着执行通道保持，注册，呼叫连接的步骤。但是在注册和 SIP 呼叫的过程中，需要对 SIP 消息做一些修改，把 SIP 消息中携带的用于建立呼叫连接和媒体通信的地址信息改为 NAT 绑定的公网地址和端口，因为私网地址在公网上是不可路由的。呼叫连接建立之后就可以直接建立点对点的媒体流传输。如果不存在 NAT 就不需要执行 NAT 穿越的流程，用户可以直接建立连接并进行 RTP 媒体流的通信。由于 STUN 自身不支持对称 NAT 的穿越，所以如果是位于对称 NAT 之后，则启动 TURN 客户端。TURN 客户端通过与公网上的 TURN 服务器通信，获得在 TURN 服务器上为它分配的地址以及端口，后续的 SIP 信令和 RTP 媒体流都通过 TURN 服务器的中继完成对对称 NAT 的穿越。与 STUN 一样，在注册和 SIP 呼叫的过程中，也需要依据 TURN 消息对 SIP 消息做一些修改。建立 RTP 通道，即可外部主机进行通信。需要结束通信时，就结束 SIP 信令和 RTP 通道。何时结束通信，则是由 SIP 用户代理决定。

由于项目中已完成 STUN 功能的实现，所以本文仅关注 TURN 部分。

4.2 TURN 服务器的实现

4.2.1 服务器软件执行过程

图 4.2 为 TURN 服务器的流程图。包括对非 TURN 的 UDP 数据包的处理以及对 TURN 数据包的处理。

TURN 服务器首先设置其对外端口，默认设置为 3478，可以通过配置文件进行修改。创建活动列表，初始情况下活动表为空，在后续接收到某类消息的触发下，会添加或修改活动表。程序启动时设置定时器，并判断是否超时以及如果超时，为哪一类超时。如果为状态转换超时，表示状态转换没有完成，则主动将外部活动地址设置为下一个外部地址，并发送 Set Active Destination 应答，表示接受 TURN 客户端设置活动目的地址的请求。这样，TURN 客户端与外部用户之间后续的数据包都不需要再进行封装，而是直接进行转发。这样将包的中继从应用层移至传输层处理，降低服务器负荷。如果是尚处于活动状态的列表超时，则更新定时器，允许它继续为活动状态。否则，删除绑定。

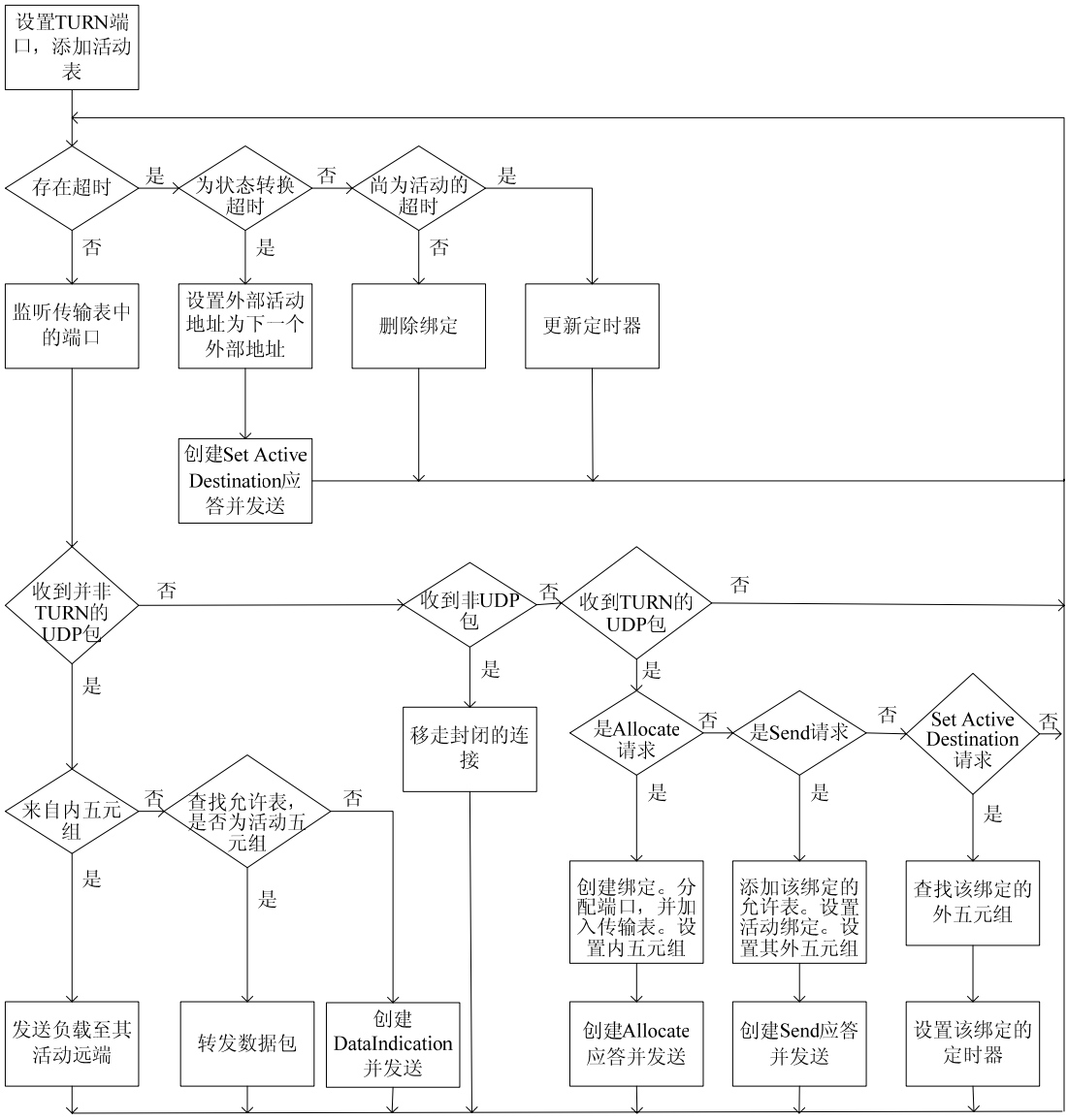

图 4.2 TURN 服务器流程图

如果不存在超时，则监听其传输列表上的端口。TURN 服务器当且仅当外部客户端在 TURN 服务器上建立了许可权限的时候，TURN 服务器才会接受来自外部客户端的 UDP 数据包或者 TCP 连接。客户端是通过 TURN Server 向指定的传输地址发送数据包来建立许可权限的。本文仅考虑 TURN 的 UDP 实现，故忽略 TCP 连接。对接收到的包，服务器判断其类型。TURN 服务器对包类型的判断在下面会有描述。

如果接收到的包不是 UDP 包，则直接结束连接。如果接收到的包是 UDP 包但不是 TURN 类型，则查看源地址是否与内部五元组匹配。如果匹配，则表示这个包来自 TURN 客户端，并且该客户端之前已经向 TURN 服务器发送过 Send 请求，并成功登记，所以此时将负载发送该客户端对应的活动远端。如果不是来自内部五元

组，则查找允许表，查看它是否为活动五元组。如果是，则表示这个包来自与外部用户，并且该外部用户已通过内部 TURN 客户端的通信允许，于是 TURN 服务器直接将数据包转发至内部 TURN 客户端。如果源地址与活动五元组不匹配，则将数据包放在 Data Indication 中，发送给内部 TURN 客户端。如果接收到的包是 UDP 包且为 TURN 类型，则依据类型不同进行相应处理，其详细过程在下面有描述。

1 对包的判断

TURN 服务器对包的判断是通过检测包的总字节数以及第 25 到 28 字节。如果包长小于 28 字节，则它不是 TURN 请求。如果长于 28 字节，则服务器检查第 25 到 28 字节，如果这些字节等于 MAGIC-COOKIE，则为 TURN 请求，否则，为需要被中继的数据包。如图 4.3 所示。

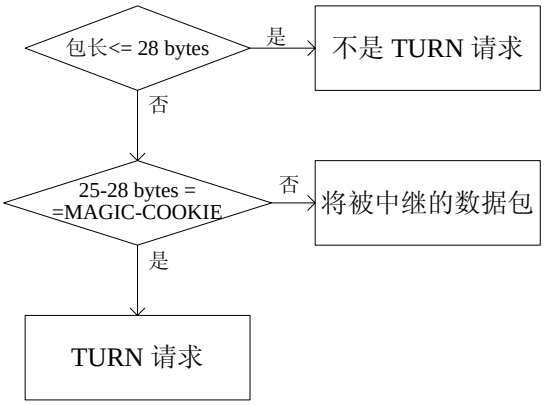

图 4.3 包类型判断

2 对 Allocate 请求的处理

Allocate 请求用来获取 IP 地址和端口，客户端可以使用这个 IP 地址和端口来接受来自外部网络任何主机的 UDP 或 TCP 数据包，甚至当客户端位于对称 NAT 之后。为此，TURN 服务器分配了一个本地传输地址，并且在分配响应中返回给客户端。TURN 服务器预先配置好了一些规则来判断从外部客户端收到的数据包是否可以被转发给客户端。这是通过一系列的地址和可选端口来标识外部客户端是否被允许了。这个地址集合是通过来自客户端的一系列 Send 请求建立的。

服务器收到分配请求时的行为取决于收到的包是初始 Allocate 请求还是后续 Allocate 请求。初始 Allocate 请求的源、目的传输地址和以存在的一个内部五元组的源、目的地址相匹配。后续请求的源、目的地址和已存在的内部五元组的远端和本地传输地址是不匹配的。如图 4.4 所示。

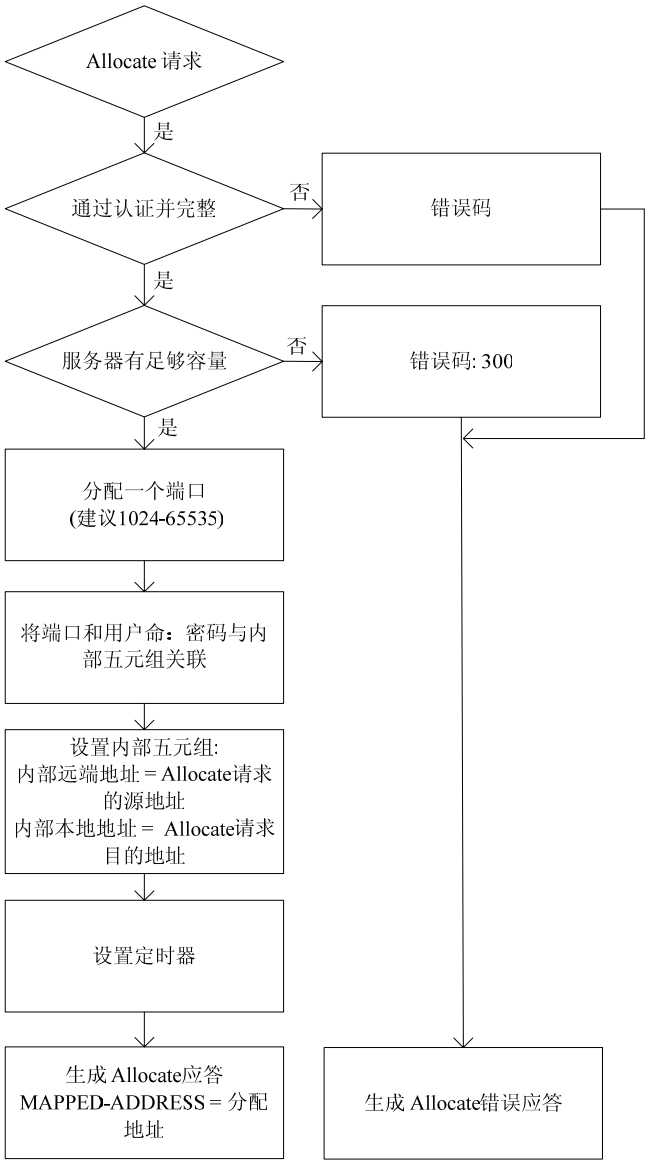

图 4.4 服务器对 Allocate 请求的处理

3 对 Send 请求的处理

为了在服务器中建立许可权限和向外部客户端发送数据，客户端完成分配事务之后就会发送 Send 请求。服务器一旦收到 Send 请求，就会校验它的源、目的地址和一个已存在的分配对应的内部五元组的远端、本地传输地址是否匹配。如果不匹配，服务器必须产生一个 Send 错误响应。之后服务器对请求进行认证，认证之后，要检查请求是否正确无误，请求中应该包括 DESTIANTION-ADDRESS 属性和 DATA 属性。否则应该拒绝这个请求。在 UDP 的情况下，服务器建立一个 UDP 包，其中的负载就是 DATA 属性中的内容，服务器把分配的传输地址作为源地址，DESTIANTION-ADDRESS 属性中的内容作为目的地址。然后发送该 UDP 数据包。

需要注意的是数据包可能需要重传，但是服务器不负责数据包的重传。这应该由客户端产生一个新的 Send 请求来负责完成。如果客户端以前没有发送过数据到这个外部的目的地址和端口，那么一个五元组就被实例化了。本地和远端传输地址分别对应的是 UDP 数据包的源、目的地址。

如果 UDP 包或 TCP 数据发送没有错误，服务器将产生一个 Send 响应，响应将从请求的目的地址被发送，发送到请求的源地址和端口。服务器将会把 DESTINATION-ADDRESS 属性对应的 IP 地址加入到这个分配对应的权限表中。该过程如图 4.5 所示。

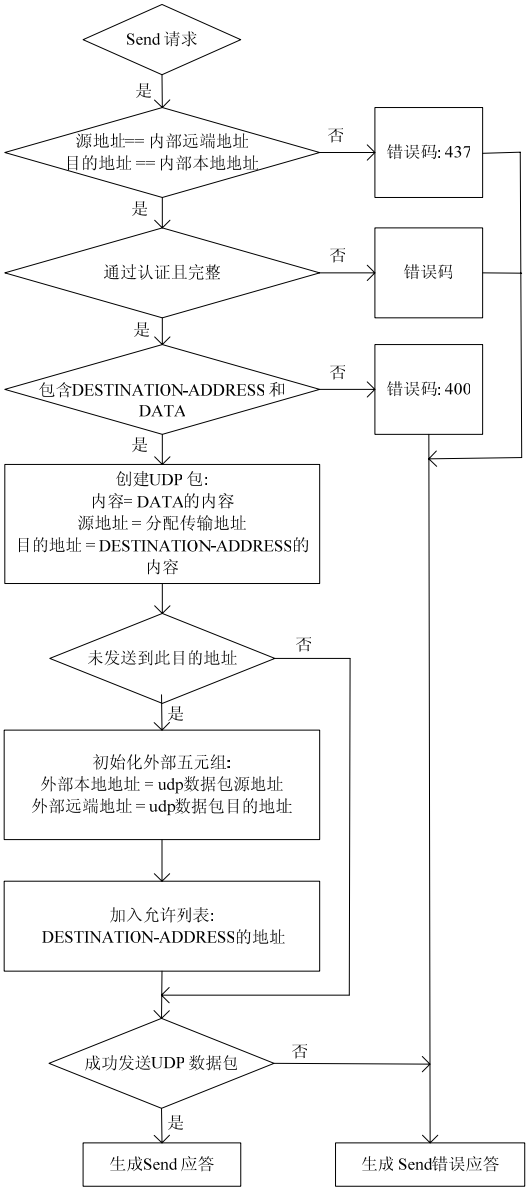

图 4.5 服务器对 Send 请求的处理

4 对 Set Active Destination 请求的处理

客户端使用 Set Active Destination 请求来决定一个外部五元组，这个外部五元组被用来作为所有来自 TURN 客户端的非 TURN 数据的转发目的地。而且，所有来自外部客户端的数据不用封装成 Indication 消息就可以被转发到客户端。活动目的地被初始化为空，它是明确地由 Set Active Destination 请求来设置的。服务器一旦识别出请求是 Set Active Destination 请求，服务器首先校验它的源地址和目的地址分别和存在的一个分配的内部五元组的内部远端地址、内部传输地址相匹配，如果没有匹配的分配存在，服务器产生一个 Send 错误应答。然后，服务器对请求进行认证，认证成功之后，要检查请求中的属性，请求中应该包括 DESTINATION-ADDRESS 属性，否则应该拒绝这个请求，并发送错误响应。

如果已经存在一个活动的五元组，并且五元组的外部远端传输地址和 DESTIANTION-ADDRESS 属性中的目的地址相匹配的，那么这个请求基本上不做任何动作，服务器必须产生一个 Set Active Destination 应答。如果存在一个活跃的五元组，但是外部传输地址和属性中的目的地址并不匹配，这个请求就要求对目的地做出改变。服务器首先在已存在的外部五元组中寻找外部远端传输地址和属性中的目的地址相匹配的，如果没有发现，请求就用 440 响应拒绝该请求。若发现有，则当前的外部活跃五元组被设置成不活跃。同时设置了一个 3 秒的定时器，服务器把自己的状态设置为“转换”状态。当定时器触发时，服务器返回到正常的操作。并且把和属性中目的地址相匹配的外部五元组设置为活跃五元组。如果没有活跃五元组，而且服务器不是处在转换状态，服务器首先在已存在的外部五元组中寻找外部远端传输地址和属性中的目的地址相匹配的，如果发现了，就把这个外部五元组设置成活跃的，并发回一个 Set Active Destination 应答，如果没有发现，请求就用 440 响应拒绝该请求。该过程如图 4.6 所示。

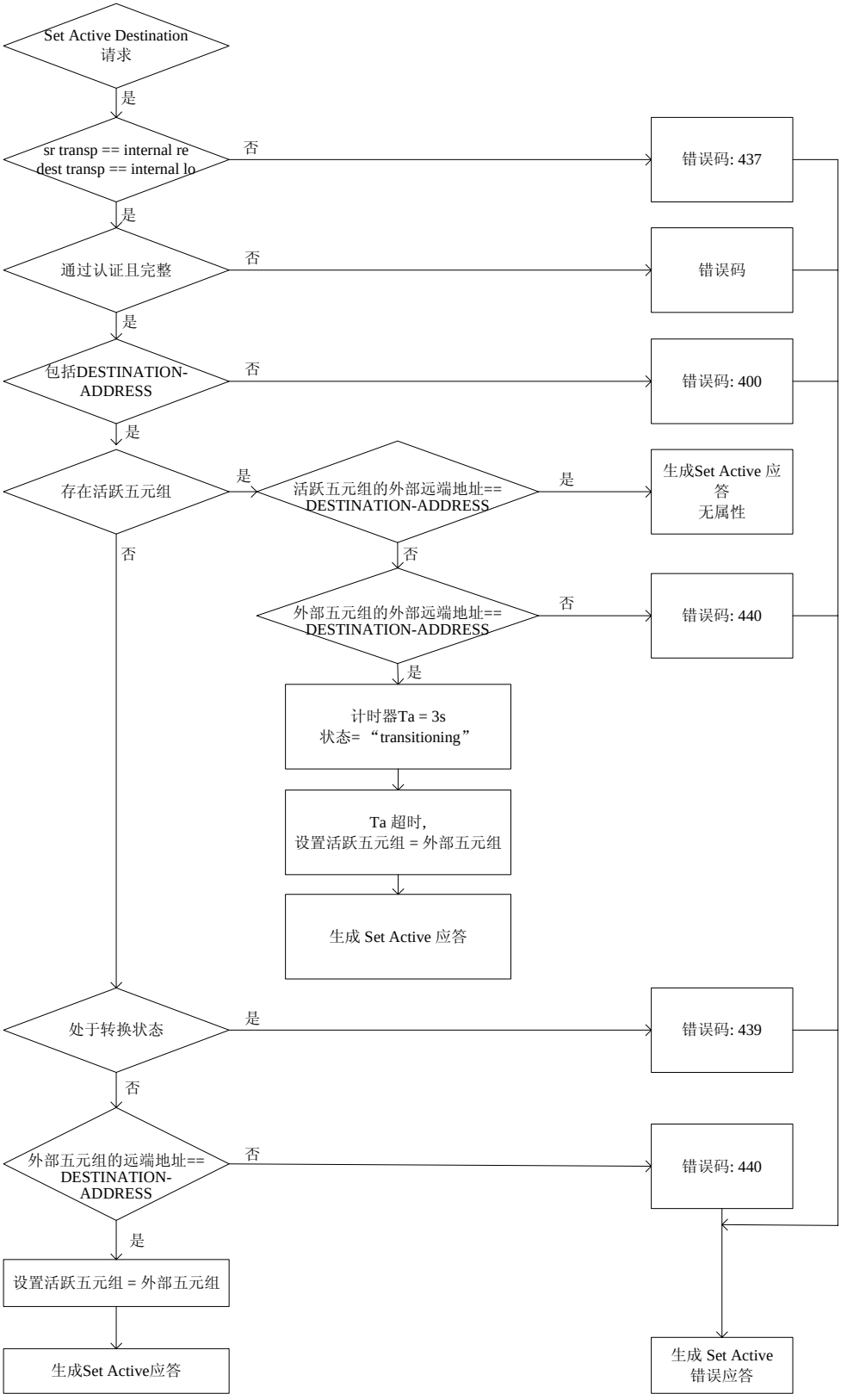

图 4.6 服务器对 Set Active Destination 请求的处理

4.2.2 类定义

此项目中，使用 C++作为编程语言，各部分使用类来实现。并且，所有的类都继承于类 Mobject，它包含一个对计数器的引用，以决定何时将对象从堆中移出。

与 TURN 消息有关的类有：TURNAttriubute、TURNMessage、MessageHeader 等。

与绑定有关的类有：TURNBinding、FiveTuple、FiveTupleContainer。

与传输有关的类有：TURNTransport、TURNTransportUDP 等。

它们之间的继承关系如图 4.7 所示。

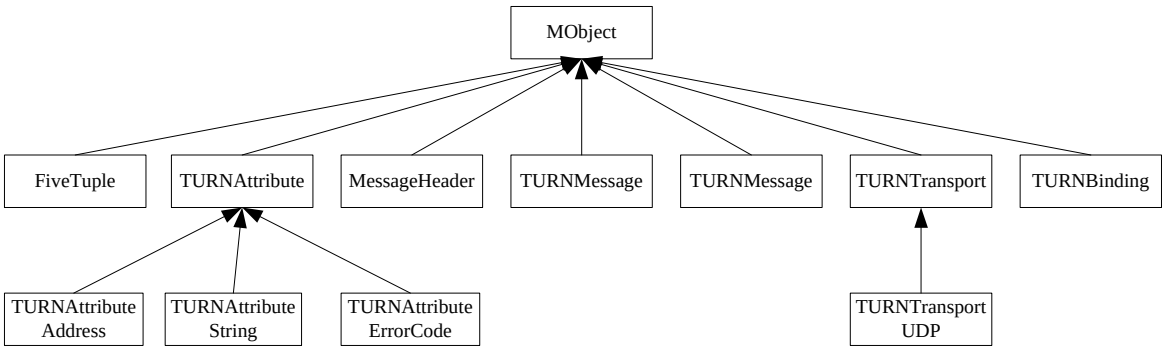

图 4.7 类的继承关系

4.3 TURN 客户端的实现

4.3.1 客户端软件执行过程

图 4.8 为 TURN 客户端的流程图。当 SIP UA 通过 STUN 交互发现自己处在对称 NAT 内时，启动 TURN 客户端。TURN 客户端返回分配的地址以及数据包供其他模块使用。

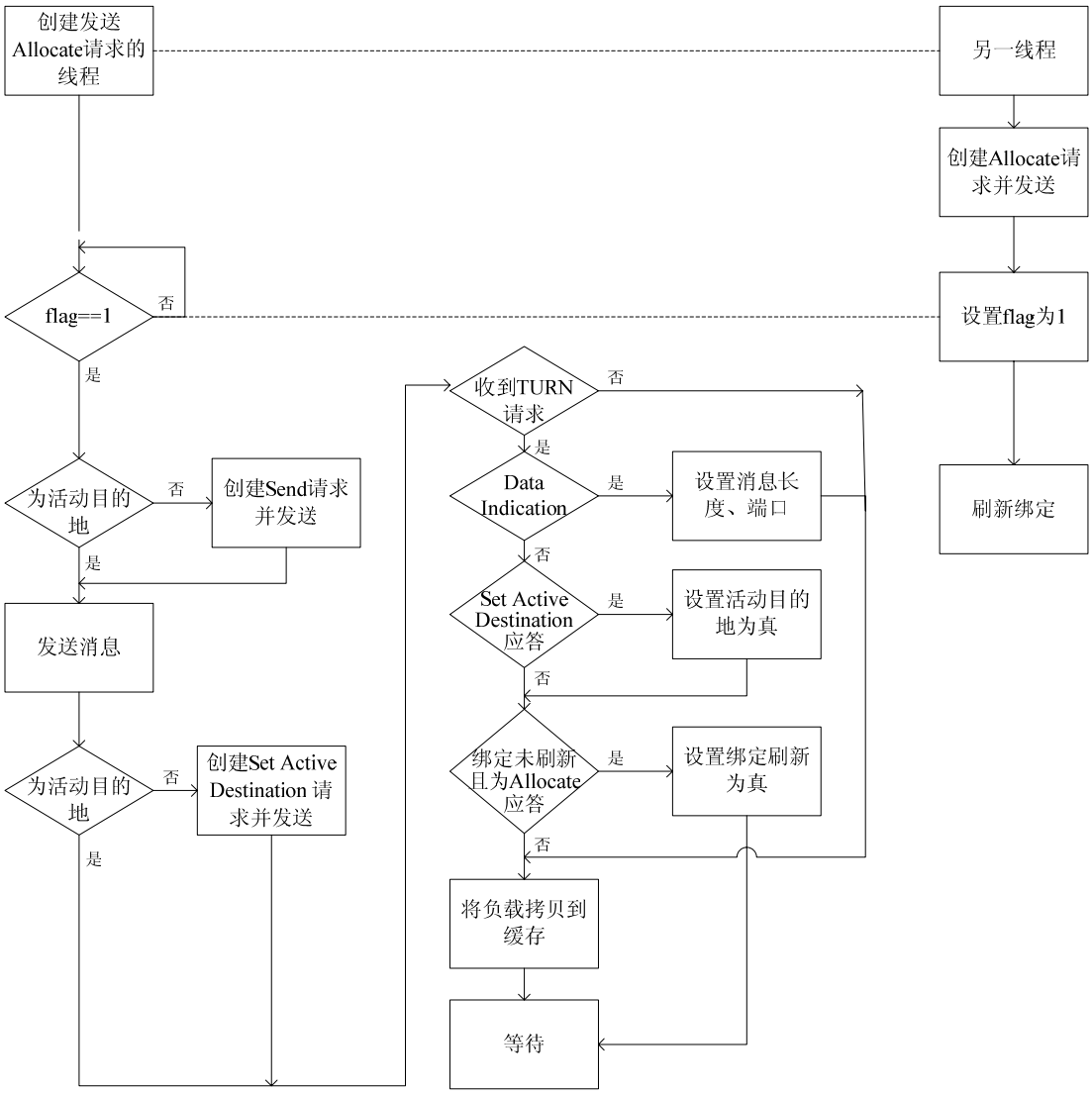

图 4.8 TURN 客户端流程图

TUNR 客户端启动后，首先发送初始 Allocate 请求，用于向 TURN 服务器索要分配的绑定地址和端口。如果对方已经是活动目的地，即当前正在通信的用户，则表示 TURN 客户端已经针对该外部用户向 TURN 服务器发送过 Send 请求和 Set Active 请求且被成功应答，可直接发送消息。否则，需要创建 Send 请求。如果尚未将正在通信的外部用户成功设置为活动目的地，则要发送的数据包作为 Send 请求的负载来发送。接着判断是否为活动目的地，如果不是则发送 Set Active Destination 请求。接着对接收到的 TURN 应答进行相应处理。如果为 Data Indication，则将负载内容拷贝到缓存，以供其他模块（SIP 模块、RTP 模块）使用。如果为 Set Active Destination 应答，则设置活动目的地址。如果为 Allocate 应答且绑定未刷新，则设置绑定刷新

为真。之后将负载均拷入缓存。

TURN 客户端为了不停地刷新与 TURN 服务器之间的绑定，需要额外起一个线程来完成这个工作。该线程不断创建 Allocate 后续请求，并携带新的生命期字段，以刷新绑定。两个线程之间通过 flag 置位来进行交互。

4.3.2 接口定义

为了使该客户端具有更好的可移植性，使用“TURN socket”代替普通的 socket，隐藏其在 TURN 方面的操作。它与普通 socket 的不同之处在于给出的 IP 以及 port 为 TURN 服务器为之分配的 IP 和 port，并通过 TURN 服务器传送数据。而 TURN 服务器的地址以及用户名、密码则通过配置文件传入。

客户端使用 C++作为编程语言开发，为方便移植，提供用 C 封装的接口。接口函数如下所示。

```
void turn_start(char *turnServer_Addr , char* turn_username , char * turn_passwd , int turn_lifetime);
```

功能：向 TURN 服务器发送绑定请求，并获得分配的地址和端口

参数：

char *turnServer_Addr，TURN 服务器的 ip 地址

char* turn_username，在 TURN 服务器上使用的用户名

char * turn_passwd，在 TURN 服务器上使用的密码

int turn_lifetime，TURN 刷新绑定时间

返回值：无

```
int turn_sendto ( int s , const void * msg , int len , unsigned int flags , const struct sockaddr * to , int tolen ) ;
```

功能：通过 TURN 服务器向外部主机发送数据

参数：

int s，已建好连线的 socket

const void * msg，指向欲连线的数据内容

int len，数据长度

unsigned int flags，一般设 0

const struct sockaddr * to，指定欲传送的网络地址

int tolen，sockaddr 的结果长度

返回值：成功则返回实际传送出去的字符数，失败返回－1

```
int turn_recvfrom(int s , void *buf , int len , unsigned int flags , struct sockaddr * from , int *fromlen);
```

功能：通过 TURN 服务器从外部主机接收数据，并把数据存到由参数 buf 指向的内

存空间

参数：

int s，已建好连线的 socket

void * buf，指向存放数据的内存空间

int len，可接收数据的最大长度

unsigned int flags， 一般设 0

struct sockaddr * from，欲传送的网络地址

int *fromlen，sockaddr 的结构长度

返回值：成功则返回接收到的字符数，失败则返回-1

```
int turn_recv(int s , void *buf , int len , unsigned int flags);
```

功能：通过 TURN 服务器从外部主机接收数据，并把数据存到由参数 buf 指向的内存空间

参数：

int s，已建好连线的 socket

void * buf，指向存放数据的内存空间

unsigned int flags，一般设 0

返回值：成功则返回接收到的字符数，失败则返回-1

4.4 系统消息流程图

4.4.1 整体消息流程图

图 4.9 为整个系统整体的消息流程图，包括 STUN 客户端与 STUN 服务器之间 STUN 消息的交互，TURN 客户端与 TURN 服务器之间 TURN 消息的交互，以及 UA（User Agent，用户代理）之间 SIP 消息的交互和 RTP 消息的交互。

为了获知是否存在 NAT 以及 NAT 的类型，UA client A 会首先向 STUN 服务器发送请求。如果获知位于 NAT 之内，则开始与 TURN 服务器进行交互。

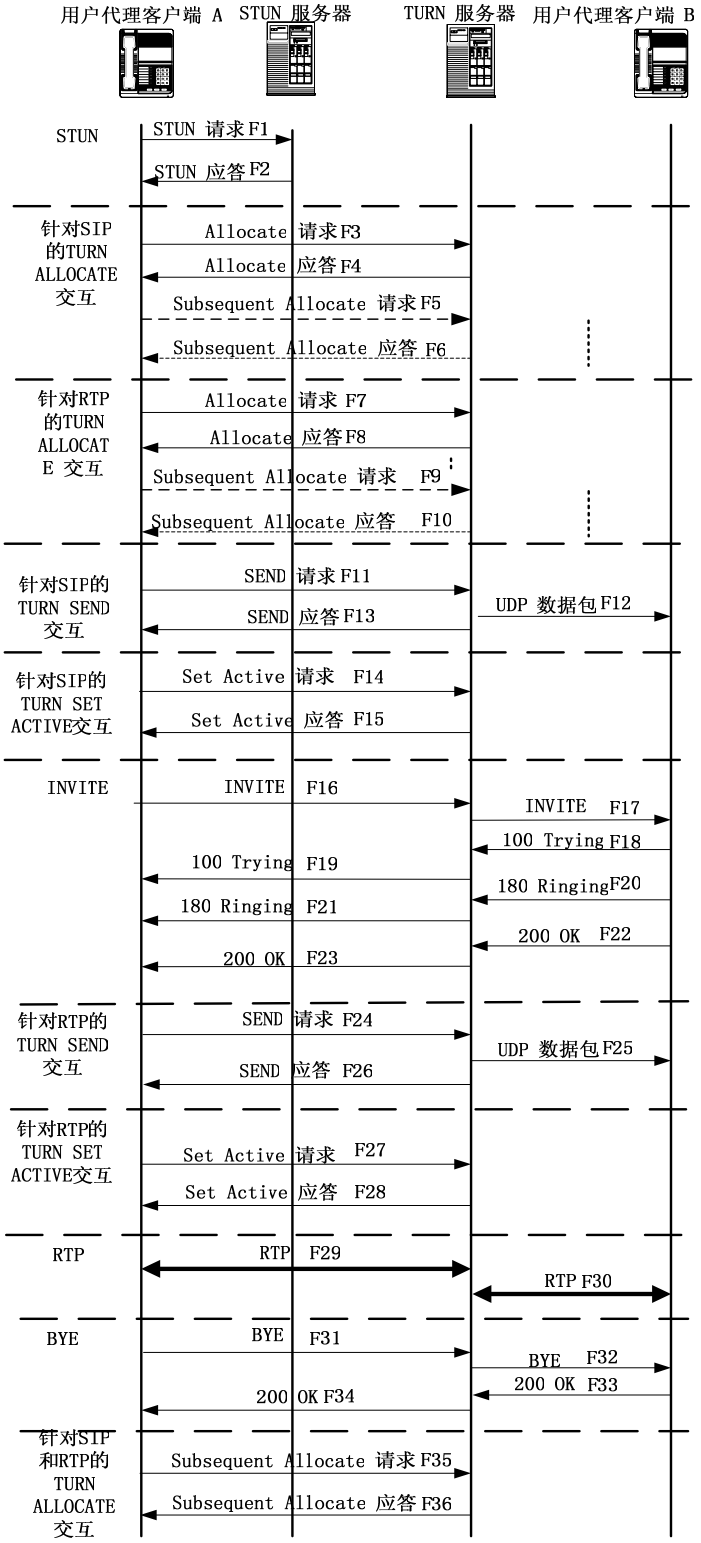

图 4.9 整体消息流程图

为了与 NAT 之外的 UA client B 之间完成 SIP 信令的正常通信，UA client A 先向

TURN 服务器依次发送针对 SIP 和 RTP 的 Allocate 请求，要求服务器为 client A 分配用于信令传输和媒体流传输的端口。在端口被成功分配之后，A 向 TURN 服务器发送以 UA client B 的 ip 地址和 SIP 端口为目的地的 Send 请求。在 TURN 服务器向 A 返回成功的 Send 应答后，A 通过 TURN 服务器穿越对称 NAT，向 B 发送 INVITE 请求以建立 SIP 会话。在收到 200 OK 成功应答后，A 向 TURN 服务器发送以 B 的 ip 地址和 RTP 端口为目的地的 Send 请求，接着发送同样目的地址的 Set Active 请求，以激活与 B 之间免封装的 RTP 交互。在此期间，A 仍要不停地向 TURN 服务器发送 subsequent Allocate 请求，以维持绑定。在发送 BYE 消息后，A 与 B 之间结束 SIP 以及 RTP 会话。最后，A 向 TURN 发送 lifetime 为 0 的 subsequent Allocate 请求，以移除绑定。

由于 SIP 会话部分和 STUN 部分已经实现，故此文主要关注 TURN 部分。该部分的详细消息流程图会在 4.4.2 节单独介绍。

4.4.2 TURN 消息流程图

图 4.10 为整个系统的 TURN 消息流程，这里假设 UA 处于对称 NAT 之内。

TURN 客户端向 TURN 服务器的公共 TURN 端口发送 Allocate 请求，TURN 服务器收到请求后，为此客户端在服务器上分配一个空闲端口，并建立内五元组，包含 Allocate 请求的源地址和目的地址等。返回 Allocate 应答，包含分配的端口号。

然后 TURN 客户端发送 Send 请求，包含将要发送的消息以及消息的接收者（即外部客户端）地址。TURN 服务器会寻找已有的内五元组，如果找到会建立外五元组，包含 TURN 服务器为 TURN 客户端分配的端口以及消息接收者地址。并且会将消息接收者地址放入允许表中。TURN 服务器将消息发给消息接收者，并向 TURN 客户端返回 Send 应答。这段时间之内，外部客户端发送给 TURN 客户端的消息被 TURN 服务器的封装成 Data Indication 之后，再发送给 TURN 客户端。而 TURN 客户端则通过发送 Send 请求间接地向外部客户端发送消息。

在 TURN 客户端向 TURN 服务器发送了 Set Active Destination 请求之后，如果 TURN 服务器找到对应的内五元组和外五元组记录，就会将消息接收者的地址设置为活动五元组。此后，外部客户端与 TURN 客户端之间就可以通过 TURN 服务器进行 UDP 包的转发。而 TURN 服务器在转发包之前，会分别查找活动五元组或者允许表以及活动五元组。

TURN 客户端向 TURN 服务器发送 lifetime 为 0 的 Allocate 请求就可以撤销绑定了。

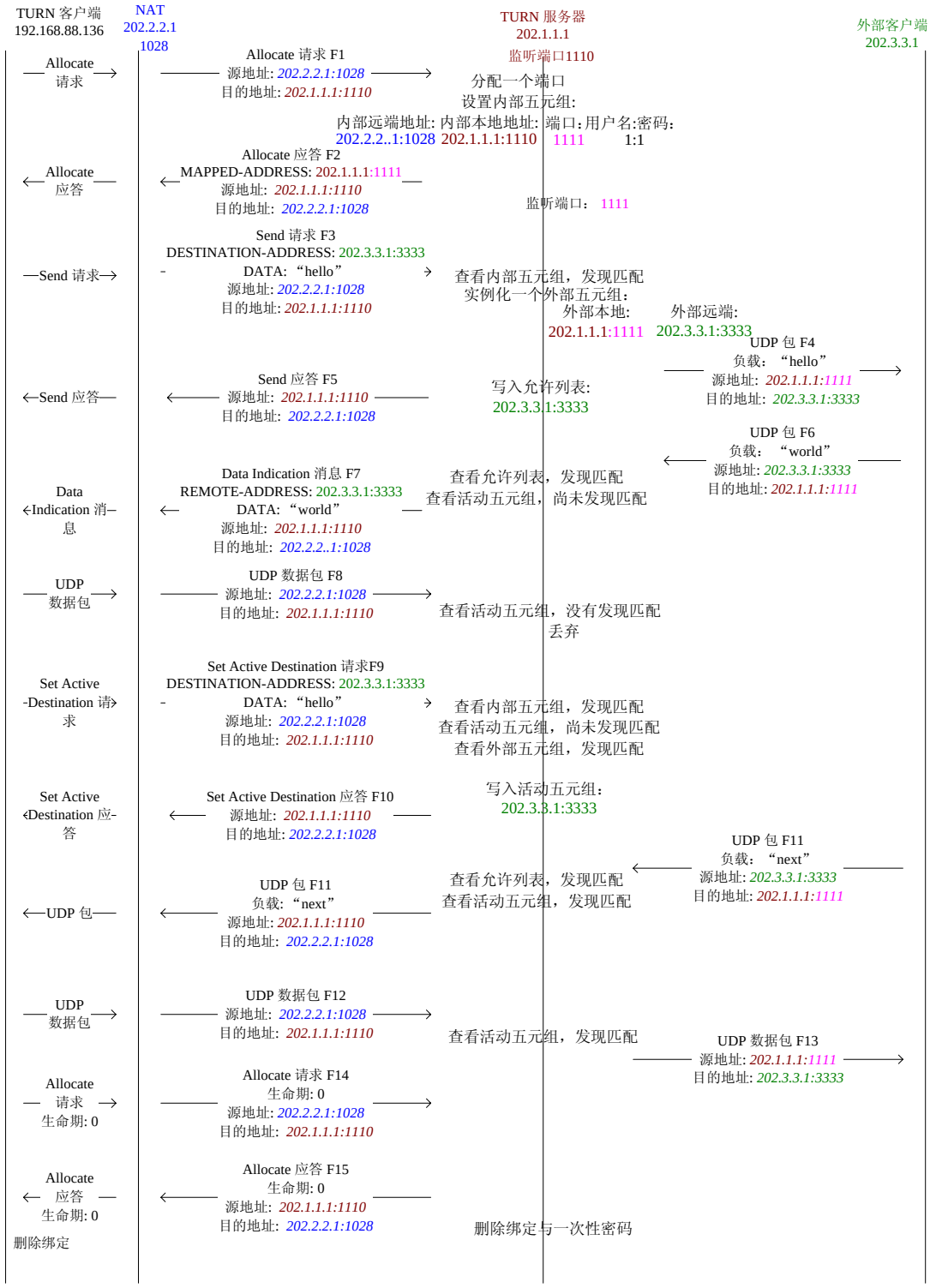

图 4.10 TURN 消息流程图

4.5 库的描述

在本文中使用到两个开源库，分别是 libmutil 和 libmnetutil。它们可以从 http://www.minisip.org/develop.html 上下载。

Libmutil 是由 minisip 提供的免费库。它由 C++语言实现，主要包含两个类：MObject 和 MRef。MObject 类包含一个引用计数器，用来决定何时从堆中移走对象。MRef 是一个模板类，使用引用计数器来与 MObject 联系。只能使用 MObject 来实例化 Mref。TURN 消息均继承自 MObject。

libmnetutil 是由 minisip 提供的免费库。它由 C++语言实现。它定义了类 IPAddress 和类 Socket， 这两者会被 TURN 传输部分以及 TURN IP 部分使用。

4.6 本章小结

本章结合第三章的设计方案，详细介绍了 TURN 模块的实现过程，主要包括 TURN 客户端的实现、TURN 服务器的实现。其次详细描述了本方案中设计的支持 TURN 协议的 SIP 终端穿越 NAT 的工作流程。使用 C++作为编程语言，对各部分进行实现。此处仅简要介绍重要的类以及封装的接口函数。

5 测试结果

为了对本文设计和实现的 VoIP 网关中 SIP 穿越 NAT 方案的功能、可靠性和适用性等方面做出较为全面的测试，本章搭建了系统测试环境，并结合穿越方案本身的特点，设计了详细的测试方案，最后对测试结果进行了分析和总结。测试结果表明本文提出的基于 TURN 协议的方案可以满足 VoIP 网关中对称 NAT 穿越的需求。

5.1 系统测试环境

本文设计的方案是在基于嵌入式 Linux 平台的 VoIP 网关中实现的，所以本文的测试是以 VoIP 网关作为平台，对 SIP 穿越 NAT 方案进行测试。图 5.1 是进行测试所需要的软硬件设备和系统测试的网络拓扑图。

硬件环境：

(1) VoIP 硬件测试板

(2) Intel Pentium IV PC 机 3 台，其中一台作为作为开发平台及硬件模拟平台。
芯片：Pentium IV 2G，内存：512M

(3) 模拟电话 1 部

(4) Hub 1 台和网线若干

软件环境：

(1) WindowsXP，RedHat Linux 9.0 操作系统

(2) SIP 网络电话 SIPPS V2.1.3.25

(3) 网络数据包捕获工具 Ethereal 0.9.4

(4) Linux 下 C/C++编译工具（如 GCC 等）

(5) 交叉编译环境（Mipsel GCC 交叉编译器）

(6) Windows 下的 source insight 3.0。

(7) Windows 下的 UltraEdit-32 。

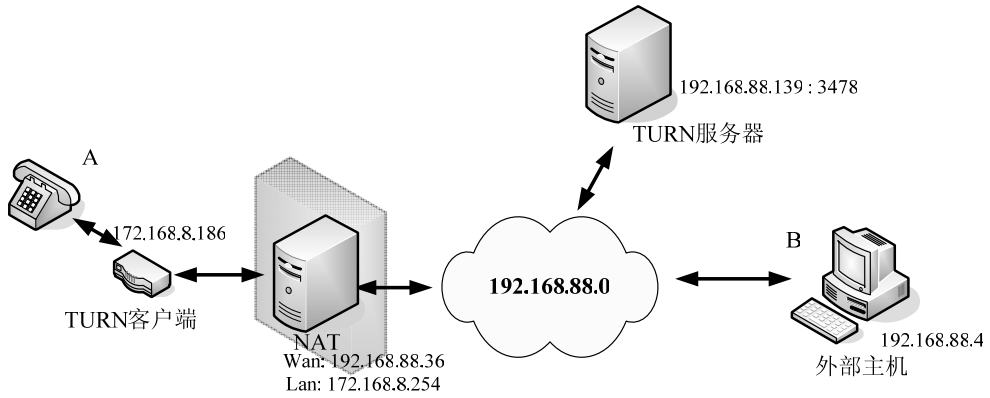

图 5.1 测试环境

使用装有双网卡的主机作为对称 NAT，集成有 VoIP 功能的板子上运行有 TURN 客户端程序，并连在 NAT 的 LAN 下，NAT 的 WAN 和 TURN 服务器、外部主机均连在 192.168.88.0 网段。

NAT 的 WAN IP 为 192.168.88.36，LAN IP 为 172.168.8.254，将其配置成对称 NAT，WAN 口接 192.168.88.0 网段。

TURN 服务器的 IP 为 192.168.88.139，使用 3478 作为 TURN 服务的对外端口。

TURN 客户端的 IP 为 172.168.8.186，通过 hub 连在 NAT 的 LAN 口。它的 FXS 端口上连接有普通电话机。

外部主机的 IP 为 192.168.88.4，运行着一个基于 PC 的 SIP 电话－－SIPPS。

5.2 系统测试方案及结果

下面给出两种典型的测试案例：

1. 内网用户呼叫外网用户（内网 TURN 客户端 A 呼叫 SIP 电话 B）

测试步骤如下：

(1) 内网用户 A 111@172.168.8.186 呼叫 NAT 外用户 B 222@192.168.88.4，等待回铃音；

(2) SIP 电话 B 222@192.168.88.4 开始振铃，摘机后呼叫连接建立；

(3) SIP 电话 B 摘机后能够正常通话，话音质量良好；

(4) SIP 电话 B 222@192.168.88.4 挂机，语音中断，通话完毕。

2. 外网用户呼叫内网用户（SIPPS 电话 B 呼叫内网用户 A）

测试步骤如下：

(1) 公网用户 SIP 电话 B 222@192.168.88.4 呼叫内网用户 A 111@172.168.8.186 ，

等待回铃音；

(2) 内网用户 A 开始振铃，摘机后呼叫连接建立；

(3) A 摘机后能够正常通话，话音质量良好；

(4) 公网用户 SIP 电话 B SIPPS 222@192.168.88.4 挂机，语音中断，通话完毕。

测试过程中，对称 NAT 之内的电话与外部 SIPPS 多次通话，均正常。在 TURN 服务器的终端显示上，可以看到有数据的转发过程。

5.3 结果分析

从结果中可以看到，通过 TURN 服务器的中转，TURN 客户端与外部主机之间通信正常。说明通过 TURN 服务器/客户机成功完成了对称 NAT 的穿越。

方案无需扩展协议，无需升级 NAT 设备。但由于 TURN 协议自身原因，从测试中可看到，因所有数据包均经过 TURN 服务器的中转，使得语音通信存在轻微延迟。实际过程中，会遇到多种类型 NAT 的穿越，将 TURN 方式与 STUN 方式结合使用，当 NAT 类型为非对称型 NAT 时使用 STUN 方式进行穿越，则可减小 TURN 服务器的负荷。

5.4 本章小结

本章以 VoIP 网关为基础，搭建了 SIP 穿越 NAT 方案的测试环境，主要包括软硬件环境的配置，同时设计了 NAT 穿越方案的测试网络拓扑图。根据搭建的测试环境和方案测试网络拓扑图，给出典型的测试案例，对本文设计的 SIP 穿越 NAT 方案进行了详细的测试。最后对对测试结果进行了总结、分析，指出了本方案的优点所在，也指出了本方案的存在的不足之处。

6 结束语

6.1 研究成果

此次工作成果主要是解决在 IP 上承载语音和视频的协议的控制通道/媒体通道无法穿越对称 NAT 与公网进行互通的问题；与 STUN 结合，可以穿越所有类型的 NAT；集成到板子上，可以解决 VoIP 对所有 NAT 的穿越，用途更广。

本文设计并实现了对称 NAT 的一种穿越方式，主要使用 TURN 技术。当该应用启动时，应用程序中的 TURN 客户端向它的服务器发送请求，开始彼此之间的交互。利用 TURN 服务器的中转，TURN 客户端穿越对称 NAT，主机进行正常通信。而与外部与 STUN 结合，可以穿越所有类型的 NAT。依据 TURN 草案，本文设计并实现了 TURN 服务器/客户端。并测试系统的正确性以及稳定性。结果显示此次的设计和实现的合理性。

6.2 发展方向

NAT 穿越问题还有很多的解决方案值得研究，本文只是对基于 TURN 协议的这种方案做了一些研究工作。文章所提出的方案可以满足穿越对称型 NAT 的问题的需要，但是本方案也还存在着一些不足和需要完善的地方，下面是一些值得进一步探讨和研究的方向：

(1) 考虑到时间以及实现难度，本文仅实现 TURN 的 UDP 方式，而并未实现其 TCP 方式。但作为安全性考虑，使用 TCP 方式会更好。

(2) 由于采用的是集中式的方式，本方案中需要架设专门的 TURN 服务器，客户端都需要和服务器通信，可以考虑把 TURN 服务器也嵌入到 SIP 终端中，由位于公网上的 SIP 终端来充当 TURN 服务器，而不需要架设专门的服务器。因此研究基于 P2P 技术的分布式 NAT 穿越方式是 SIP 穿越 NAT 领域的一个重要方向。

(3) 加快研究和普及 IPv6 技术，从根本上解决 IPv4 地址资源的缺乏问题。

致 谢

值此论文结束之际，回顾两年来的硕士之路，我要衷心感谢所有帮助过我的老师、同学、朋友和亲人，并借此机会对他们致以最诚挚的谢意！

首先感谢我的导师杜旭副教授在整个研究生阶段对我的悉心指导和严格要求。他严谨务实的治学态度、渊博的学识、积极进取的创新意识、豁达乐观的学者风范，给了我很多启示。他以丰富的科研经验，给了我毫无保留的指导，促进了我对专业知识的学习和掌握，让我受益匪浅。从他身上，我不仅学到了严谨的学习工作作风、扎实的科研精神，还学到了更多的为人处世之道。

感谢左冬红老师、黄佳庆老师和余江老师，感谢他们在我研究生学习期间给予我的无私帮助，也感谢他们在我研究学习遇到困难时给予我的及时指导。他们扎实的专业功底和管理能力是我学习的楷模，同时，也感谢他们为我的论文提出的大量宝贵意见。

感谢实验室朝夕与共，共同奋斗的师兄弟姐妹们，特别是 VoIP 组的全体成员。能够在这样一个由大家共同构筑的幸福温馨的集体里面完成两年的学习，是我的幸运。感谢熊建强师兄、刘杉杉师姐等在实验室的课题开发中对我的支持。感谢我的室友，两年来共同进退的日子让我难忘，我会永远记得和怀念同大家在一起的朝朝暮暮、点点滴滴。也祝大家在今后的人生道路上一帆风顺！

最后，感谢我的父母，没有他们的默默支持，我不可能完成如此漫长和艰苦的学业。他们的言传身教使我受益非浅，在此向他们致以最衷心的感谢和祝福！

闵江

二零零八年五月于喻园

参考文献

[1] Jon Postel. Internet Protocol. IETF RFC 791, 1981

[2] K. Egevang, P. Francis. The IP Network Address Translator (NAT), RFC 1631. May 1994

[3] Paul E. Jones. Draft H.323v4: Paket Based Multimedia Communication Systems [S]. ITU-T Recommendation H.323, November 2000.

[4] 李爽，赵琛. SIP 和 H.323 协议介绍及比较. 中国数据通信, 2001, 12 : 35~38

[5] Jonathab Rosenberg, Henning Schulzrinne, Gonzalo Camarillo et al. SIP: Session Initiation Protocol. IETF RFC 3261, June 2002

[6] 徐静华，voip 中 nat 穿越解决方案的设计与实现: [硕士学位论文]. 保存地点: 华中科技大学图书馆, 2005

[7] 熊建强,黄佳庆,熊志强 等. 一种 SIP 穿越 NAT 方案的设计与实现. 计算机工程与科学, 2007,29(8). 102~104

[8] J. Rosenberg，R. Mahy and C. Huitema. Traversal Using Relay NAT (TURN), Internet-Draft. September 9, 2005

[9] B. Biggs. A SIP Application Level Gateway for Network Address Translation, Internet-Draft. March 2000

[10] 佟欣哲. NGN 私网用户 VOIP 的 NAT 穿越. 广东通信技术, 2004, 12: 11~14

[11] Miller B A, Nixon T, Tai c, et al. Home Networking with Universal Plug and Play. IEEE Communications Magazine. 2001, 2:12

[12] 赵志峰，杨永康，仇佩亮. UPnP 在多媒体通信穿越 NAT 中的应用. 电信科学, 2006, 6: 10~14

[13] M. Stiemerling, J. Quittek, T. Taylor. Middlebox Communication (MIDCOM) Protocol Semantics, RFC 3989. March 2008

[14] B. Ford, P. Srisuresh, D. Kegel. Peer-to-Peer (P2P) communication across middleboxes. IETF Draft, October 2003.

[15] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), RFC 3489. March 2003

[16] P. Srisuresh, M. Holdrege. IP Network Address Translator (NAT) Terminology and

Considerations, RFC 2663. August 1999

[17]司端锋, 韩心慧, 龙勤, 潘爱民 SIP 标准中的核心技术与研究进展, 软件学报. 2005, 16(2): 239~250

[18]王瑞章, 田小华. VoIP 穿过 NAT 及防火墙的挑战. 通讯世界, 2003, 12: 68~69

[19]H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, RFC 1889. January 1996

[20]何永龙, 林浒, 雷为民. 一种 SIP NAT 应用网关的设计与实现. 小型微型计算机系统, 2002, 23(8): 913~916

[21]Srisuresh, P. and K. Egevang. Traditional IP Network Address Translator. IETF RFC 3022, January 2001

[22]吴伟. NGN 业务穿越 NAT/FW 方案的解决方案. 电信技术, 2004, 12: 15~18

[23]Geoff Huston. Anatomy: A Look Inside Network Address Translators. The Internet Protocol Journal, 2004, 7(3): 1~32

[24]何宝宏. 浅析 NAT 的类型. 电信网技术, 2004, 8: 4~7

[25]刘怀兰, 匡松柏, 陈光亮, 黄若宏. 基于隧道机制实现 H.323 中防火墙和 NAT 的穿越. 华中科技大学学报(自然科学版), 2005, 33(6): 12~13

[26]许先斌, 万庆. 一种解决 SIP NAT 的方案的设计与实现. 计算机应用, 2004, 24(4): 83~85

[27]Jon Postel. User Datagram Protocol. IETF: RFC 768, August 1980

[28]陈建华. VoIP 穿透 FW/ NAT 的方案探讨. 中国有线电视, 2004, 22: 17~20

[29]彭湘凯, NAT 技术及其在网络安全中的应用. 四川师范大学学报(自然科学版), 2001, 24(5): 542~545

[30]梅黎. 基于软交换的 NAT/防火墙穿透技术研究. 现代电信科技, 2005, 1: 28~30

[31]张波, 胡瑞敏, 边学工. 一种实现 SIP 穿越 NAT 的新方案. 网络与通信, 2005, 31: 119~121

[32]杜吉友, 张艳霞, 董德存. 穿越防火墙/NAT 的 SIP 通信研究. 中国数据通信, 2004, 2: 71~74

[33]张仙伟, 王琨. 会话的初始化协议. 电子科技, 2005 ,4. 38~40

[34]谢刚. NGN 中的 SIP 技术. 江西通信科技, 2005, 3: 10~12

[35]许苏明, 王忠民. SIP 协议及其应用. 世界电信, 2002, 10: 45~48

[36]林铮. 软交换中的关键技术 SIP. 通信世界, 2002, 7: 38~39

[37]沈波, 张顺颐, 沈苏彬. 会话发起协 SIP 的分析和研究. 数据通信, 2001, 4: 15~18

[38]陈燕，龚建荣. SIP 协议的内容及其基本网络结构. 广东通信技术, 2005, 4: 69~71

[39]王敏. VoIP 网关中的 SIP 协议研究及实现: [硕士学位论文]. 武汉: 华中科技大学, 2004.

[40]陈华林，盛翊智. SIP 协议中的媒体协商.广东通信技术, 2005, 8: 71~73

[41]乐燕群，程时端. IP 电话信令协议—SIP. 电信技术, 2000, 2: 24~25

[42]李伟章. SIP 协议浅谈. 电信技术, 2004, 10：84~86