北京邮电大学

硕士学位论文

VNC穿越NAT相关技术的研究

姓名：庄霄

申请学位级别：硕士

专业：计算机应用技术

指导教师：邓中亮

20090210

VNC 穿越 NAT 相关技术的研究

摘 要

VNC (Virtual Network Computing，虚拟网络计算)是 AT&T 剑桥实验室所研发的屏幕分享与远端操作工具，它支持多种操作系统，可以用于实现远程办公，远程技术支持等多种应用。随着互联网的迅速发展，互联网用户的迅速增多而 IPv6 尚未普及，IP 地址匮乏的问题日益严峻，NAT（Network Address Translate, 网络地址转换)成为了解决这一问题的普遍手段。但是 NAT 也带来另外一个问题，外部网络终端无法直接访问 NAT 内的终端,这使得位于 NAT 后的 VNC 客户端与服务端之间很难进行正常通信和工作。如何让 VNC 穿越 NAT 正常工作成为了 VNC 应用中的一个重要问题。

本文针对 NAT 的原理进行了深入的研究。通过对现有的 NAT 穿越技术的分析和对比，提出了一种可以综合解决 VNC 跨越 NAT 的方案。由于 VNC 通过 UDP 协议来进行一些控制信息的传输，而一些实际数据的，如屏幕状态则需要通过 TCP 协议来传输。所以我们的 VNC 穿越 NAT 方案包括了 UDP 穿越和 TCP 穿越两个部分。其中 UDP 穿越部分使用了目前比较成熟，应用非常广泛的 STUN 协议，而 TCP 穿越部分由于涉及到三次握手和状态装换要比 UDP 复杂的多，目前业内还没有一个被广泛认可的方案，通过对几种 TCP 穿越技术的分析比较我们最终选择了 NUSS 小组的 STUNT 协议为基础来进行开发。但是基于 STUNT 实现的 TCP 穿越方法依然着不少问题，本文针对此这些问题进行了更深入的研究，对 STUNT 模型进行了改进，达到更了好的 NAT 穿越的成功率。最后我们加入了端口预测算法，来解决 STUN 和 STUNT 协议中都没有涉及到的对称性 NAT 穿越的问题。

本文的最后提出了一套完整的 VNC 穿越 NAT 解决方案，详细描述了整个方案的架构及工作流程。并在北航实验室以及石景山区市政府等地部署了实验环境对其进行了实验测试，以实验数据证明了方案的可行性。

关键词： VNC, NAT 穿越，STUN,STUNT

NAT-Traversal Technology for VNC

Abstract

VNC (Virtual Network Computing) is a screen sharing and remote operation tools developed by the AT & T Laboratories in Cambridge, it supports multiple operating systems, it can be used to variety of applications such as remote office, remote technical support, With the rapid development of the Internet, the number of Internet user are growing fast. Otherwise Ipv6 is not yet universal,the problem of IP address shortage is becoming more and more serious., IP address shortage problem is becoming increasingly severe, NAT (Network Address Translate) has become the universal solution to this problem. But NAT has also brought some issue, the external network terminal can not directly access to the terminal behind NAT, it is couse the communications between VNC client and server which is located behind NAT became very difficult. How to make VNC work through NAT become an important issue for VNC applications.

In this paper, we do a in-depth research with NAT and conducted things. Through analysis and comparison with the existent NAT traversal technology, rise a NAT traversal comprehensive solution for VNC. Because VNC use both UDP protocol and TCP protocol to transmit the infomation. This solution includes UDP traversal and TCP traversal two parts. UDP traversal part used the STUN protocol which is relatively sophisticated, and used widely, but the TCP traversal part, as which requires to three-way handsake and status changes, is much more complexed than UDP one. at present the industry has not yet a has widely recognized solution, by analysis and comparison, through several TCP traversal technical we finally chose the STUNT protocal for Nuss group to carry out basic development. However, the TCP traversal solution based on STUNT still have many problems, this paper did a in-depth research for these issues and improve the STUNT model to achieve better success rate. Finally, we add the port prediction algorithm, to solve the Symmetric NAT traversal problem which is not mentioned from STUN and STUNT protocol.

Finally this paper gave detailed description for the overall program architecture and work flow, builded an experimental environment for test. experimental result is demanstrated that the program is feasible.

Key words: VNC, NAT- Traversal， STUN,STUNT

中英文术语对照表

术语说明：

ISP: Internet Service Provider 因特网服务提供商

VNC: VNC (Virtual Network Computing)是虚拟网络计算机的缩写

NAT Network Address Translators 网络地址转换

STUN:Simple Traversal of UDP Through NAT UDP 对 NAT 的简单穿越方式

TURN: (Traversal Using Relay NAT)通过转发方式穿越 NAT

MIDCOM: the Middlebox Communications protocol 中间盒协议

NAT/ALG: 网络地址转换/应用层网管

NUSS : NUTSS 是康乃尔大学提出一个穿透不同 NAT 直接连接 TCP 的项目

STUNT: Simple Traversal of UDP through Nats and TCP

NATBlaster: 一种 TCP 穿越方法

TTL:Time To Live 生命周期

NB:Independent 独立端口映射

NB:Address 地址端口映射

NB:Address and Port 地址端口映射

NB:Session 连接端口映射

EF:Independent 独立过滤

EF:Address 地址过滤

EF:Port 端口过滤

EF:Address and Port 地址端口过滤

DNAT:Destination NAT 目的网络地址翻译

SNAT: Source NAT 源网络的地址翻译

独创性（或创新性）声明

本人声明所呈交的论文是本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中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他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。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：王雪　　日期：3.17

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。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磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：王雪　　日期：3.17

导师签名：邵中亮　　日期：3.17

第一章 绪论

1.1 课题背景

1.1.1 VNC 技术概述

VNC[1] (Virtual Network Computing，虚拟网络计算)是 AT&T 剑桥实验室所研发的屏幕分享与远端操作工具，它支持多种操作系统，可以支持远程办公，远程技术支持等多种应用。通过一个简单的客户端软件("VNC viewer")你就可以在互联网的任何一个位置，浏览并控制另一台安装了服务端("VNC server")的电脑。

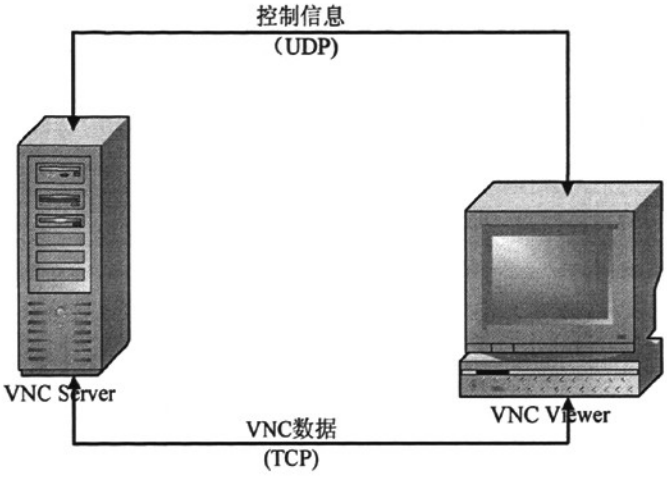


图 1-1 VNC 工作原理

如图 1-1 所示其中服务端软件负责将鼠标键盘动作，屏幕变化等信息压缩然后通过网络传送到客户端软件。客户端就可以看到安装了服务端的电脑上的情况，服务端软件还可以将从客户端发送过来的各种鼠标键盘控制信息，转换为实际的鼠标键盘动作，这样客户端就可以通过 VNC 来控制安装了服务端的电脑。

控制信息:包括连接请求，抓屏频率，压缩参数由于实时性要求不高，主要通过 UDP 协议来进行传输。

VNC 数据：包括键盘鼠标动作，屏幕变化则通过 TCP 协议来传输。

VNC 与 Windows 自带的简单的远程控制方法不同。首先 VNC 是操作系统无关的， 可以很简单的跨越平台。比如你可以通过一台安装了 Windows Vista 的电脑来操作另一台安装着 Linux 的电脑。 也可以通过一台安装着 Linux 的电脑来

操作控制安装着 Windows 系统的电脑。 甚至在没有安装客户端软件的电脑上你也可以通过诸如 IE, Netscape 等支持 Java 脚本的浏览器来使用。其次 VNC 在传输质量，传输速度上有更多的选择和优化。而且可以实现一对多，甚至多对一的操作方式。

随着网络的高度发展，电脑的管理及技术支持的需要，VNC 及其相关控制技术越来越引起人们的关注。目前 VNC 技术有一下主要有以下几种应用：

1)远程办公

通过远程控制功能我们可以轻松的实现远程办公，这种远程的办公方式新颖，轻松，从某种方面来说可以提高员工的工作效率和工作兴趣。

2)远程技术支持

通常，远距离的技术支持必须依赖技术人员和用户之间的电话交流来进行，这种交流既耗时又容易出错。但是有了远程控制技术，技术人员就可以远程控制用户的电脑，就像直接操作本地电脑一样，只需要用户的简单帮助就可以得到该机器存在的问题的第一手材料，很快就可以找到问题的所在，并加以解决。

3）远程交流

利用远程技术，商业公司可以实现和用户的远程交流，采用交互式的教学模式，通过实际操作来培训用户，使用户从技术支持专业人员那里学习示例知识变得十分容易。而教师和学生之间也可以利用这种远程控制技术实现教学问题的交流，学生可以不用见到老师，就得到老师手把手的辅导和讲授。学生还可以直接在电脑中进行习题的演算和求解，在此过程中，教师能够轻松看到学生的解题思路和步骤，并加以实时的指导。

4）远程维护和管理

网络管理员或者普通用户可以通过远程控制技术为远端的电脑安装和配置软件、下载并安装软件修补程序、配置应用程序和进行系统软件设置。

由于 VNC 通过 UDP 协议来进行一些控制信息的传输，而一些实际数据的，如屏幕状态则需要通过 TCP 协议来传输。

1.1.2 NAT 技术概述

NAT(Network Address Translators)网络地址转换[2]:网络地址转换是在 IP 地址日益缺乏的情况下产生的,它的主要目的就是为了能够地址重用。NAT 是一个 IETF 标准,它可以将局域网中的内部地址节点翻译成合法的 IP 地址在 Internet 上使用(即把 IP 包内的地址域用合法的 IP 地址来替换),或者把一个 IP 地址转换成某个局域网节点的地址,从而可以帮助网络超越地址的限制,合理的

安排网络中共有 Internet 地址和私有 IP 地址的使用。使用这种通过地址转换访问不同网段信息的 NAT 技术，专用网络上的多台 PC(使用专用地址段，例如 10.0.x.x, 192.168.x.x,172.x.x.x)共享单个全局路由的IPv4地址，减少了所需的 IPv4 地址。 NAT 与防火墙或代理服务器不同，通常的 NAT 功能会被集成到路由器，防火墙，单独的NAT设备中。

最开始NAT是运行在路由器上的一个功能模块，他的产生基于如下事实：一个似有网络(域)中的节点只有很少的节点需要与外网连接。那么这个子网中其实只有少数的节点需要全球唯一的IP地址，其他的节点的IP地址应该是可以重用的。因此，基本的 NAT 实现的功能很简单，在子网内使用一个保留的 IP 子网段，这些 IP 队外是不可见的。子网内只有少数一些 IP 地址可以对应到真正全球唯一的 IP 地址。如果这些节点需要访问外部网络，哪么基本 NAT 就负责这个节点的子网内 IP 转化为一个全球唯一的 IP 然后发送出去。基本的 NAT 会改变 IP 包中的原 IP 地址，但是不会改变 IP 包中的端口。

现在广泛使用的 NAT 也叫做 NAPT(Network Adress/Port Translator)，从名称上我们可以看得出，NAPT 不但会改变。经过这个 NAT 设备的 IP 数据报的 IP 地址，还会改变 IP 数据报的 TCP/UDP 端口。现在大部分的 NAT 都是基于 NAPT 的，而基于基本 NAT 的设备已经很少见了。

1.1.3 穿越 NAT 问题的提出

VNC 在工作时需要在两台电脑之间建立 TCP/UDP 的连接，并传递数据。首先 NAT 的地址端口映射会使 NAT 外的电脑无法获知 NAT 内电脑的实际 IP 地址，其次 NAT 会阻挡来自 NAT 外所发起的连接，所有不是由网内电脑发起通信数据将会被拦截。 这使得 NAT 后的两台电脑无法顺利建立连接 也就无法正常通信。如何让 VNC 穿越 NAT 正常工作成为了 VNC 应用中的一个重要问题。

1.2 国内外研究现状

目前国内外针对解决 NAT 穿越问题提出了很多方法,UDP 方面有虚拟局域网 VPN, 代理 Full Proxy, NAT/ALG , MIDCOM,STUN,TURN 等，其中 STUN 最为成熟。

NAT/ALG[3]：为了实现 NAT 穿越,最早出现了将 ALG 放到 NAT 中,即网络地址转换/应用层网管(NAT/ALG)方式去解决 NAT 穿越问题。但这个方式可扩展性不强,每增加一种应用都需要对相应的 ALG 设备进行升级。

MIDCOM[4]与 NAT/ALG 不同的是,MIDCOM(the Middlebox Communications protocol 中间盒协议)的框架结构是采用可信的第三方(MIDCOM Agent)对 Middlebox(NAT/FW)进行控制的机制转移到外部的 MIDCOM Agent 上,因此应用协议对 Middlebox 是透明的。MIDCOM 技术的优势在于可扩展性强,新增应用只需对 MIDCOM Agent 进行扩展,而不会导致 MIDBOX 重新升级。随着 MIDCOM 协议的不断成熟和发展,该方式将获得越来越多的应用前景。

STUN[5]: (Simple Traversal of UDP Through NAT [RFC3489]) 即 UDP 对 NAT 的简单穿越方式。通过端口映射和在 NAT 上“打洞的方式”实现穿越， STUN 是目前应用最广泛的 NAT 穿越方法，有着很多成功的案例，但是 STUN 的缺陷也很明显，无法实现 TCP 的穿越以及对称 NAT 穿越。

TURN[6]: (Traversal Using Relay NAT)通过转发方式穿越NAT.STUN 只负责协助穿越，两台电脑开始通信后就可以脱离 STUN 服务器了。而 TURN 方式则负责转发所有信息。这使得 TURN 可以应付对称 NAT。但是这使 TURN 服务器的负担过重,增大了包的延迟,效率会降低,同时丢包率也升高了

代理[7]: 代理方式是在局测进行修改应用层内容,即由代理服务器来提供地址转换功能。代理服务器需要同时对信令和媒体进行中继转发,工作效率和转发速度会受影响,而且还不可避免的增加了数据包的时延和丢包可能。

但基于 TCP 的 NAT 穿越目前业内还没有一个被广泛认可的方案。

1.3 本文主要工作及组织结构

1.3.1 主要工作

针对课题的研究内容及项目的实际需要，本文作者在论文期间主要做了以下工作：

对网络地址转换（NAT）技术基本原理，分类方法进行了深入的研究，分析了不同种类的 NAT 对 VNC 造成的影响。以及 NAT 映射类型,终端过滤对 VNC 的造成的影响。这部分体现在本文的第二章

深入研究了了现有的 NAT 穿越技术包，并对这些技术的特点，实现原理，以及优缺点进行了分析和比较。其中 UDP 穿越部分使用了目前比较成熟，应用非常广泛的 STUN 协议，而 TCP 穿越部分由于涉及到三次握手和状态装换要比 UDP 复杂的多，目前业内还没有一个被广泛认可的方案，通过对几种 TCP 穿越技术的分析比较我们最终选择了 NUSS 小组的 STUNT 协议为基础来进行开发。但是基于 STUNT 实现的 TCP 穿越方法依然着不少问题，本文针对此这些问题进行了更深入的研究，对 STUNT 模型进行了改进，达到了更好的 NAT 穿越的成

功率。最后我们加入了端口预测算法，来解决 STUN 和 STUNT 协议中都没有涉及到的对称性 NAT 穿越的问题。这部分体现在第三章。

提出了一套完整的 VNC 穿越 NAT 解决方案，并详细描述了整体方案的构架及工作流程。这部分体现在第四章。

最后部署了实验环境，并在北航实验室以及石景山区市政府等地区进行了实验，实验数据表明本方案的 NAT 穿越成功率可以达到 95%。这部分体现在第五章。

1.3.2 组织结构

第一章 简要介绍了论文的课题背景、课题内容及课题意义，然后介绍了作者在攻读硕士学位期间，论文所做的主要工作以及论文结构。

第二章 主要介绍了研究网络地址转换（NAT）技术。

第三章 对 NAT 穿越技术的研究，通过分析以及比较选取了 STUN 和 STUNT 作为 VNC 穿越 NAT 的解决方案。对 STUNT 模型进行了改进，提高了穿越成功率，加入了端口预测来穿越对称 NAT,

第四章 提出了一套完整的 VNC 穿越 NAT 解决方案，并详细描述了整体方案的构架及工作流程。这部分体现在第四章。。

第五章 部署了实验环境并对方案进行了实验,根据实验数据进行了分析。

第六章 对本文进行了总结并展望了未来发展的趋势。

第二章 网络地址转换（NAT）技术

2.1 NAT 基本原理

随着社会信息化建设的发展，越来越多的单位建立了自己的网络系统，并且对网络的建设不仅仅局限于局域网的建设，同时需要将局域网和 Internet 或者和自己行业的广域网络相连，如政府机关的公共信息网等。要进入 Internet 必须要有合法的 IP 地址，只有拥有合法 IP 地址的信息包才能进入广域网，实现信息的交换和资源的共享。可以通过向 ISP(Internet Service Provider,因特网服务提供商)申请合法的 IP 地址来解决这个问题。但是，合法的 IP 地址资源非常有限，能申请到足够多的合法 IP 地址非常困难，并且需要高昂的费用。尽管 IPV6 协议可以解决地址资源空间不足的问题，但是从 IPv4 到 IPV6 的过渡不可能在很短的时间完成，所以很多企业更愿意通过使用未经注册的保留 IP 地址来解决这一问题。如果一个用户正在计划着将网络联入 Internet，而用户只能从网络服务提供商那里获得很少注册的合法 IP 地址，NAT(Network Address Translation,网络地址转换)可以帮助解决一部分问题。NAT 通过将写在 IP 报头上的未经注册的保留地址替换为合法的、已获注册的 IP 地址来解决地址冲突的问题[22]。

NAT 顾名思义就是实现了网络地址的翻译，其实现的核心是把内部网络中数据报文的地址翻译为外部合法地址的数据报文并向外部网络发送，而在收到外部数抿报文后，再翻译为内部地址并向内部网绪发送。NAT 的实质就是动态维护一个映射表，用来把内部的地址(IP 地址，端口)映射到合法的外部地址上去。当内部有数据报文时，若真实目标地址不在内部网络（说明需要地址函译），NAT 系统就查找该映射表。如果目标地址已存在，则直接翻译转发报文，否则分配新的地址资源，生成新的表项后再翻译转发报文。收到外部报文后，同样也查表，如果没有查到，直接丢弃该报文，否则进行地址翻译并向内部网络转发。当然，该表项必须动态维护，超过一定时间还未继续使用的外部地址会被系统回收。

2.2 NAT 的分类

从不同的角度来看，可以将 NAT 分为不同类型的 NAT。下面从三个角度的分类来了解 NAT 的具体情况。即从 NAT 的映射地址的范围，NAT 对修改的数据包的目的地址还是源地址，以及 NAT 对建立的映射情况向内转发的严格程度三个不同的角度来看，可以按下面三个标准来对 NAT 技术进行分来。

从 NAT 的映射地势的范围来看，NAT 有三种类型，NAT(staticNAT)，NAT 池(pooledNAT)和端口(PAT)[2]。其中静态 NAT 设置起来最为简单，内部网络中的每个主机都永久映射成外部网络中的某个合法地址。而 NAT 池则是在外部网络中定义了一系列的合法地址，采用动态分配的方法映射到内部网络。PAT 则是把内部地址映射到外部网络的一个 IP 地址的不同端口上。

从 NAT 对修改的数据包的目的地址还是源地址来看，还可以将 NAT 分为 DNAT[22] (Destination NAT 目的网络地址翻译)和 SNAT[23] (source NAT 源网络的地址翻译)两种类型。其中，对于 SNAT，完成向外的源网络地址翻译，一般用于私有网络地址内部的主机通过 NAT，访问外部网络，对于 DNAT，则正好相反，完成向私有网络内部的目的网络地址翻译，一般用于外部主机访问私有网络内部的主机。

目前可以把 NAT 分为 4[23]种 见图 2-1：

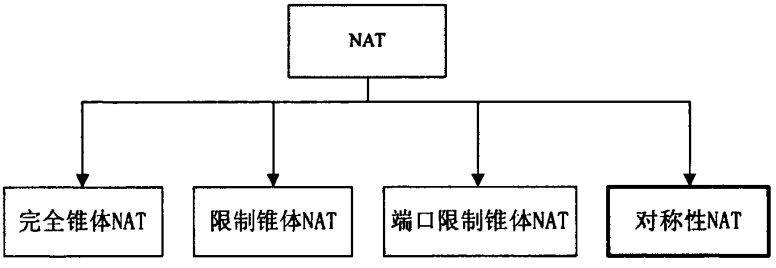


图 2-1 NAT 分类

1. 完全锥体(Full cone) NAT
2. 限制锥体(Restricted cone) NAT
3. 端口限制锥体(Port restricted cone)NAT
4. 对称性(Symmetric) NAT

一般前面 3 种可以归为一类，都称作非对称性 NAT 或者锥体 NAT。和第 4 种的区别在于前面 3 种的 NAT 映射不会根据目的地址的不同而改变，而第 4 种同步 NAT 会根据报文的目的地址的不同而改变映射。

下面将对这四种类型的 NAT 分别做一个简单的介绍：

1. 完全锥体(Full cone) NAT

完全锥体(Full cone) NAT 的情况下，NAT 的映射表已经建立起来，处于外网的任何人想访问 NAT 之后的用户只要知道内网所映射的地址就行了。所有从同一个 NAT 后 IP 地址，端口出去的访问都被映射成同一个合法地址，任何公网上想要达到 NAT 后的一个客户端，仅需要知道的是该客户端的 NAT 后的合法地址，

好发送包给客户端。

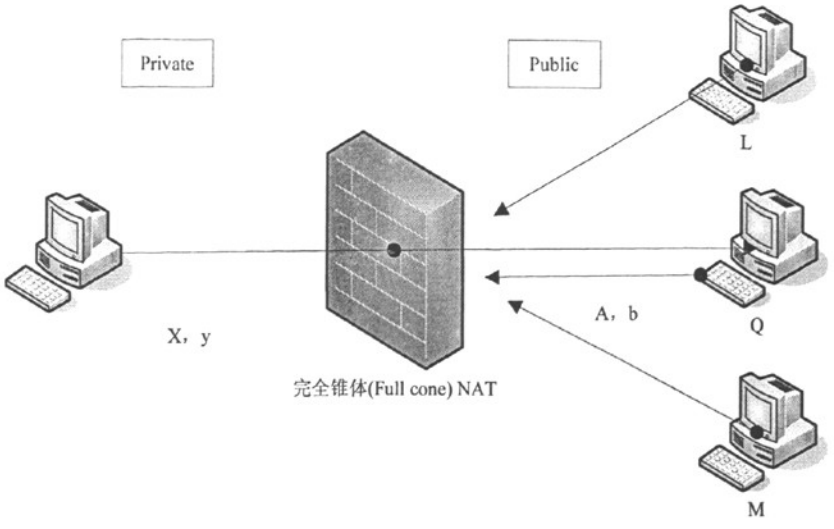


图 2-2 完全锥体(Full cone) NAT

如图 2-2 所示，NAT 后的地址（X，y）访问公网上的主机 Q，地址被映射成（A，b），公网上的其他主机（L，M）同样可以通过地址（A，b）访问 NAT 后的主机。

2. 限制锥体(Restricted cone) NAT

限制锥体(Restricted cone) NAT 的情况下，映射只有在内网的用户先发送包到外网特定的 IP 地址的情况下才建立，这样想用这个外网 IP 地址同内网通信的用户只有等到先收到内网发到这个地址的包才行。所有从同一个 NAT 后 IP 地址，端口出去的访问都被映射成同一个合法地址，但 NAT 的地址，端口对仅在 NAT 后的客户端发送数据给一个特定的外部目的地址时开放。

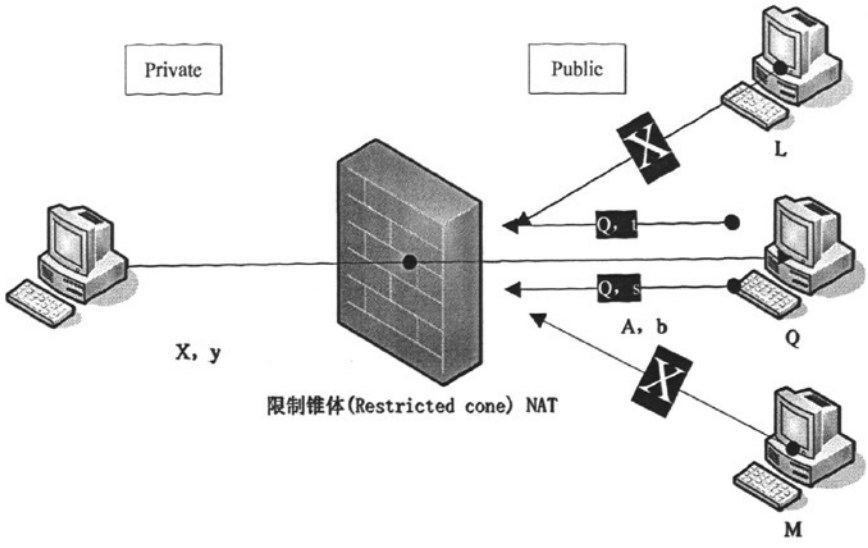


图 2-3 限制锥体(Restricted cone) NAT

如图 2-3 所示，NAT 后的地址（X，y）访问公网上的主机 Q，地址被映射成（A，b），公网上的其他主机（L，M）知道映射的合法地址（A，b）后不能访问私网内主机。只能主机地址为 Q 可以访问。

3. 端口限制锥体(Port restricted cone)NAT

端口限制锥体(Port restricted cone)NAT 的情况几乎等同于限制锥体(Restricted cone) NAT，映射只有在内网的用户先发送包到外网的特定 IP 地址和端口的情况下才建立，这样想用这个外网 IP 地址和端口同内网通信的用户只有等到先收到内网发到这个地址和端口的包才行。但会阻塞所有的包，除非客户端先告诉外部用户其主机的某个端口发送信息给他。

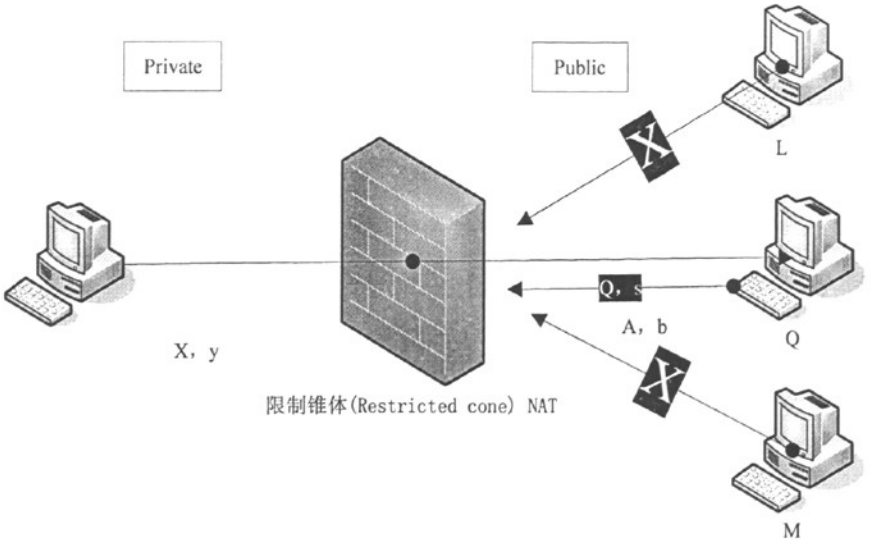


图 2-4 端口限制锥体(Port restricted cone)NAT

如图 2-4 所示，NAT 后的地址（X，y）访问公网上的主机 Q，地址被映射成（A，b），公网上的其他主机（L，M）知道映射的合法地址（A，b）后不能访问私网内主机。只有 Q 主机的 s 端口可以访问。

4. 对称性(Symmetric) NAT

对称性(Symmetric) NAT 同前面三种都不一样在于映射依赖于目的 IP 地址。也就是说同样的内网地址发到不同的外网 IP 会有不同的映射。同时还依赖与目的地址地址、端口。所有来自相同内部 IP 地址和端口到同一目的地的请求都映射到某一个相同的外部 IP 地址和端口上。加入相同的主机使用相同的源 IP 地址和端口号发送信息包到另一目的地,哪么将会映射到另一外部地址上。而且,只有当外部主机接收到内部主机的信息包后才能发送信息包回给该内部主机。

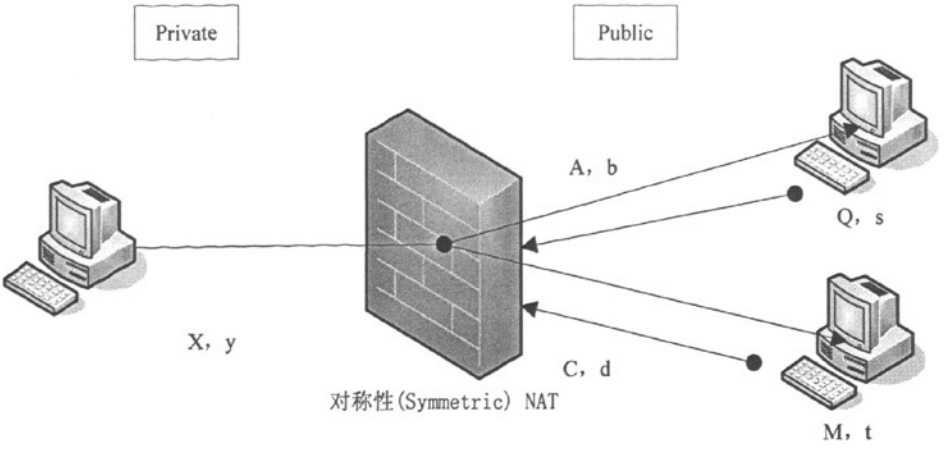


图 2-5 对称性(Symmetric) NAT

如图 2-5 所示，主机地址（X，y）访问公网地址（Q，s）NAT 映射为（A，b），但（X，y）访问公网地址（M，t）的时候，NAT 映射为（C，d）。

2.3 对称性 NAT 的映射特性

对称性 NAT 会为每一个新的会话请求建立一个新的映射地址，即使这个请求来自相同的 IP 地址和端口，但是其中的映射方式是有规律可循的，主要分为全敏感型和 IP 敏感型[18]，还有极为少数的对称性 NAT 端口分配是完全随机的，这种 NAT 的端口无法预测，幸好这种 NAT 数量极为稀少的。

搭设一个简单的实验环境，终端 A 位于内网即 NAT 后，将它的内网地址设为 192.168.1.2　端口设为 1234，终端 B 位于外网，它的 IP 地址开始时为 59.64.128.10 端口为 3456.通过终端 A 向终端 B 发送连接信息，在终端 B 就可以观察到终端 A 在 NAT 上所映射的地址了。通过改变终端 B 的地址可以观察到终端 A 在 NAT 所映射地址的变化规律。

全敏感型，即一旦数据包发送的目的地址（包括 IP 地址和端口)发生变化们就建立新的映射地址，表 2-1 给出了一个例子，

表 2-1 全敏感型对称 NAT

发起顺序	源地址	目标地址	映射地址
1	192.168.1.2：1234	59.64.128.10：3456	59.64.200.10：45678
2	192.168.1.2：1234	59.64.128.10：3457	59.64.200.10：45679

			（45678+1）
3	192.168.1.2：1234	59.64.128.11：3456	59.64.200.10：45680（45679+1）
4	192.168.1.2：1234	59.64.128.11：3459	59.64.200.10：45681（45680+1）

可以看出，映射地址的端口变化是有规律的这个增量δP往往是一个固定值，一般为1或者2.(表中为1)。

IP敏感型,即只有数据包发送的目的IP地址变化后才建立新的映射地址，表2-2给出了一个例子。

表2-2 IP敏感型对称NAT

发起顺序	源地址	目标地址	映射地址
1	192.168.1.2：1234	59.64.128.10：3456	59.64.200.10：45678
2	192.168.1.2：1234	59.64.128.10：3457	59.64.200.10：45678
3	192.168.1.2：1234	59.64.128.11：3456	59.64.200.10：45689（45678+1）
4	192.168.1.2：1234	59.64.128.11：3457	59.64.200.10：45689
5	192.168.1.2：1234	59.64.128.12：3456	59.64.200.10：45690（45679+1）

可以看出，映射地址的端口随着目的IP地址的变化而变化，并且这种变化的增量δP也是一个固定值，一般为1或者2.(表中为1)。

2.4 NAT对VNC造成的影响

NAT通过将局域网中的内部地址节点翻译成合法的IP地址在Internet上使用(即把IP包内的地址域用合法的IP地址来替换),或者把一个IP地址转换成某个局域网节点的地址,从而实现IP地址的重复使用，但这也造成处于NAT后的VNC很难进行正常通信[20]。NAT照成VNC难以正常通信的具体原因，主要体现在下面两个方面。

2.4.1 NAT映射

NAT为每个基于源和目的IP和端口的TCP连接选择一个对外映射。在某些条件下，有些NAT会重用已经存在的映射，而有些则每次都分配新的映射。NAT

映射类别就捕捉这些映射行为的不同。这对于那些企图穿透 NAT 的终端来说非常有用，因为这可以让他们基于之前的连接来预测一个（新）连接的映射地址和端口。对于 UDP 来说，我们知道，有些 NAT 为那些从同一个源地址和端口产生的所有连接，总是分配一个固定的地址和端口。而对于 TCP 来说根据 NAT 的不同可以分为几种不同的模式[8]。

只要该客户端新连接的源地址和端口与第一次连接的相同，NAT 就一直重用该映射。我们把这种行为归类为 NB:Independent。因为映射只由源地址和端口决定，而独立于目的地址和端口。对应锥体 NAT。

只要只要新连接的源地址同第一次连接时的地址匹配 NAT 就会重用该映射。这种行为被归类为 NB:Address。对应锥体 NAT。

只要新连接的源地址和目标的地址和端口同时都与第一次连接的匹配，NAT 才会重用该映射。这种行为被归类为 NB:Address and Port。对应锥体 NAT。

即使新连接的的源地址和目标的地址和端口同时都与与第一次连接的匹配 NAT 也不会重用该映射。而总是建立新连接这种行为被归类为 NB:Session 对应对称性 NAT。

VNC 要进行正常通信。首先必须要知道对方在公网上的 IP 和端口号。而 NAT 的这种映射特性使得 VNC 很难直接获取到所需要的 IP 和端口信息。

2.4.2 终端过滤

NAT 可能过滤掉到发送某个端口的进入包，如果那个端口上不存在 NAT 映射，NAT 就会强制过滤掉包。然而，如果映射已经存在，或者设备是防火墙的话，可能还需要进入包的源地址和（或）端口与之前的外出包的目的地匹配。对于 VNC 来说终端过滤的影响主要为如果两个 VNC 设备都在 NAT 后。那么发起连接的一方的 SYN 请求将会被接收方的 NAT 过滤掉而无法到达。假设接收方通过某种方式收到了发送的方的 SYN 请求，而它的 ACK 也会被发起方的 NAT 过滤掉因为这时连接并没有建立。

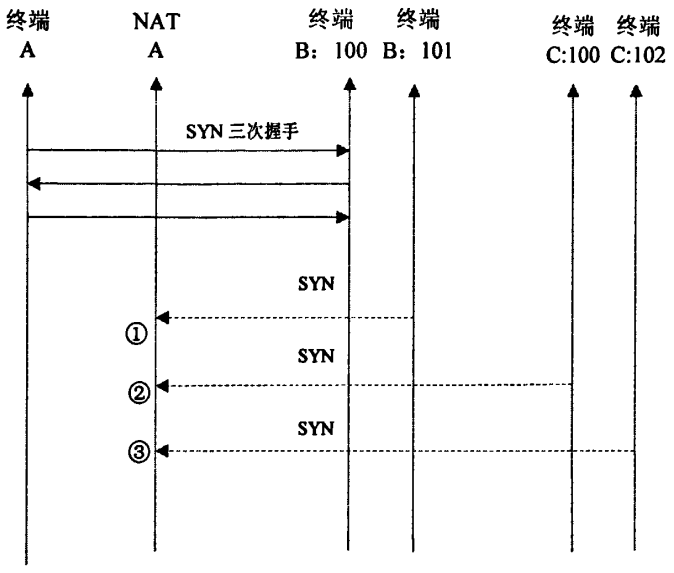


图 2-6 终端过滤示意图

图 2-6 中：

（1）是从相同 IP 不同端口发出的 SYN 消息。

（2）是从不同 IP 相同端口发出的 SYN 消息。

（3）是从不同 IP 不同端口发出的 SYN 消息。

假设由客户端向终端 B 发起一次 TCP 连接，经过三次握手后完成连接。然后通过这个连接向客户端发送 SYN 数据包。

假如 NAT 允许进入的 TCP 连接与源地址和端口无关，只要路由该请求的必要状态存在即可。我们把这种 NAT 分类为有终点过滤行为，即 EF:Independent[24]。上图中(1),(2),(3)发出的 SYN 消息都能通过 EF:Independent 型的 NAT

假如 NAT 将会过滤所有进入包，需要进入的 TCP 报的源与创建映射的（TCP）连接的目的地址和端口都匹配。这种 NAT 的终点过滤被分类为 EF:Address and Port. 上图中(1),(2),(3)发出的 SYN 消息都无法通过 EF:Address and Port 型的 NAT

假如 NAT 允许与该连接目的地有相同地址或端口的包进入，但各自会过滤从不同地址或不同端口来的包。我们把这种终点过滤分别分类为 EF:Address 和 EF:Port。上图中(1)发出的 SYN 消息都可以通过 EF:Address 型的 NAT。(2)发出的 SYN 消息都可以通过 EF:Port 型的 NAT. (3)发出的 SYN 消息则无法通过其中

的任何一种。

从上面 NAT 的分类可以看出 EF:Independent 型的 NAT 对应完全锥体(Full cone) NAT。EF:Address and Port 型的 NAT 对应对称性(Symmetric) NAT。EF:Address 型的 NAT 对应限制锥体(Restricted cone) NAT, EF:Port 型的 NAT 对应端口限制锥体(Port restricted cone)NAT。

由此我们可以得出下面的对应表，见表 2-3

表 2-3 NAT 对应关系表

	完全锥体	限制锥体	端口限制锥体	对称性
映射类型	NB: Independent	NB: Address	NB: Address and Port	NB: Session
终端过滤	EF: Independent	EF: Address	EF: Port	EF: Address and Port

2.5 本章小节

本章对网络地址转换(NAT)技术的原理进行了研究与分析，对 NAT 的分类进行了详细的解释。最后描述并分析了不同类型的 NAT 在 VNC 通信中照成的不同影响以及带来的问题。

第三章 VNC 穿越 NAT 技术研究

VNC 需要主要通过 UDP 协议来传输包括连接请求，抓屏频率，压缩参数等等传输控制信息。另外还需要通过 TCP 协议来传输包括键盘鼠标动作，屏幕变化在内 VNC 数据流。那么要使 VNC 可以正常工作我们必须同时解决 UDP 和 TCP 两方面的 NAT 穿透问题，下面将会分别对这两部分进行描述。

3.1 基于 UDP 的 NAT 穿越

3.1.1 现有 UDP 穿越 NAT 技术的比较

UDP 的 NAT 穿越技术相对比较成熟，目前比较知名的 UDP 穿越 NAT 技术有下面这些。

3.1.1.1 NAT/ALG

为了实现 NAT 穿越，最早出现了将 ALG 放到 NAT 中，即网络地址转换/应用层网管[3](NAT/ALG)方式去解决 NAT 穿越问题。但这个方式可扩展性不强，每增加一种应用都需要对相应的 ALG 设备进行升级；使得 NAT 设备的负担更加繁重，跟不上速度太快的 Client，每一个应用都需要单独的实现，影响 NAT 穿越的效率和性能；对于在使用中的大量 NAT 的升级改造工作非常困难。

3.1.1.2 MIDCOM

与 NAT/ALG 不同的是，MIDCOM[4] (the Middlebox Communications protocol 中间盒协议）的框架结构是采用可信的第三方 (MIDCOM Agent) 对 Middlebox(NAT/FW)进行控制的机，应用业务识别的只能也由 Middlebox(NAT/FW)转移到外部的 MIDCOM Agent 上，因此应用协议对 Middlebox 是透明的。MIDCOM 技术的优势在于可扩展性强，新增应用只需对 MIDCOM Agent 进行扩展，而不会导致 MIDBOX 重新升级。随着 MIDCOM 协议的不断成熟和发展，该方式将获得越来越多的应用前景。

3.1.1.3 STUN

STUN[5](Simple Traversal of UDP Through NAT [RFC3489]) 即 UDP 对 NAT 的简单穿越方式。在这种方式中,他必须包括 STUN Client----可以是运行在用户终端的系统上,如用户的计算机,也可以是运行在网络设备中,相当于一个应用程序;STUN Server---用以接手和回复 STUN Client,通常放置在公网网络中。使用 STUN 无需队现有 NAT 设备作任何改动。

STUN Client 先向 NAT 外的 STUN Server 以 UDP 数据包的形式发送请求 STUN 消息。当在公网上的 STUN Server 收到该请求消息后。STUN Server 回复响应消息,并将请求消息的源端口放在该响应消息中,即 STUN Client 用过 NAT 后被映像的外部端口。接着响应消息穿越 NAT 发送到 STUN Client, STUN Client 检查响应消息数据包,并获得其在 NAT 上对应的外部地址,然后将该外部地址填入以后护甲协议的 UDP 负载中告知通讯端,以后通讯时的 RTP 接收地址和端口位 NAT 对外的地址和端口.通过上述过程, STUN 协议可以在相应的 NAT 上预先建立媒体流的 NAT 映像表项,这样使得媒体流可以顺利穿越 NAT。

STUN 方式最大的优点是无需现有的 NAT 设备做任何改动。由于实际应用中,已有大量的 NAT,而且这些 NAT 并不支持我们需要的应用 比如 VNC, 若蚕蛹 STUN 方式无需改动 NAT。同时 STUN 方式可以在多个 NAT 串联的网络环境中使用。STUN 的缺点是不支持 TCP 连接的穿越。不支持 4 中 NAT 模型中 对称 NAT(Symmetric NAT)的穿越。 在安全性比较高的企业网中,出口 NAT 往往是对称 NAT 类型。 这就是 STUN 的局限性。

3.1.1.4 TURN

TURN[6] (Traversal Using Relay NAT)通过转发方式穿越 NAT。 TURN 和 STUN 在结构上和实现过程上都有类似的地方,同样包括了内网的 TURN Client 和公网的 TURN Server。 TURN 方式要求公网上有一个服务器,该服务器的作用是转发所有通过 NAT 的报文,TURN 服务器为每一个向 TURN 申请的内网主机分配端口,内网的主机也同样需要向服务器询问地址和端口 。

TURN 的实现过程类似于 STRUN,不同的是 STUN 方式获得的地址是 NAT 转换后对外的 IP 地址和端口,TURN 方式获得的地址是 TURN 服务器的 IP 地址和端口,即 STUN 在给用户的数据包中填写的是 NAT 转换后的 IP 地址和端口,而 TURN 则是使用 TURN Server 的 IP 地址和端口发给用户,作为内网与外网节点通讯的 IP 地址和端口。

由此可知,TURN 方式由于在对称 NAT 不会产生新的地址映像,从而可顺利实现对称NAT(Symmetric NAT)穿越, 填补了STUN无法穿越对称NAT的缺点。同时TURN方式支持基于 TCP 的应用。但是使用 TURN 方式时,内网对外网的所有数据包都必须通过TURN服务器转发.这样 TURN服务器的负担就比较重,增大了包的延迟,效率会降低,同时丢包率也升高了。

3.1.1.5 代理

代理[7] (Proxy)技术,也是一种比较流行的 NAT 穿越解决方案。代理方式是在局测进行修改应用层内容,即由代理服务器来提供地址转换功能。代理方式在网络中部署的位置比较灵活,既可以应用于私网内部,也可以应用于公网和私网边缘或者公网上. 无需对现有NAT 做任何改动,可采用普通的设备。然而,使用代理技术时每增加一种新的应用就需要对代理设备进行歇息扩展。

代理服务器需要同时对信令和媒体进行中继转发,工作效率和转发速度会受影响,而且还不可避免的增加了数据包的时延和丢包可能。

代理的优势在于不但可以应用于 UDP 穿越可以应用于 TCP 穿越甚至 HTTP 穿越

3.1.1.6 技术之间的比较

解决方案	服务器要求	现有设备改造	设备负担	安全性	对称 NAT
NAT/ALG	无	需要	大	高	可以
MIDCOM	无	需要	小	中	可以
STUN	STUN 服务器(公网 IP)	不需要	小	低	不能
TURN	TURN 服务器(公网 IP)	不需要	大	低	可以
代理	Proxy 服务器(公网 IP)	不需要	大	中	可以

表 2-1 UDP 穿越技术比较

NAT/ALG 和 MIDCOM 方案 都需要对现有的 NAT 设备进行一定程度上的改造。一方面 NAT 改造从实现角度上来讲相对困难,另一方面随着应用的升级 NAT

设备也需要随着升级和改变这大大提高了运行维护的成本。和增加了应用升级的复杂度。对于 VNC 来讲，面对的都是中小型用户，这种 NAT 设备改造是不现实的。

代理技术的泛用性最强，可以应用于穿越各种 NAT，而且不需要对现有设备进行改造。对于网页访问这种轻量级的应用可以说是最佳的选择。而 VNC 在通信的过程中需要传递大量的图像，图形数据。全部通过代理中转的话工作效率和转发速度会受影响，而且还不可避免的增加了数据包的时延和丢包可能。

TURN 技术面临的问题和代理很相似，内网对外网的所有数据包都必须通过 TURN 服务器转发，在 VNC 这种较大流量的通信方式中 TURN 服务器的负担就比较重，增大了包的延迟，效率会降低，同时丢包率也升高了

而 STUN 技术中服务器在穿越完成以后就只需要对连接的维持进行管理，而不需要参与到通信中，所有数据都由两台终端独立交流，这就解放了 STUN 服务器，使得其可以接受比较大流量的需求。而且 STUN 技术相对成熟，也有不少成功的应用 如 著名的 VOIP 软件 Skype 就是使用 STUN 技术完成的。

3.1.2 STUN 消息结构

STUN 协议中的消息，是使用 Big-endian 编码的 TLV(type-length-vlaue)字节流。所 有的 STUN 消息开始于 STUN 头，接着是 STUN 负载[9]。负载是一系列的 STUN 属性，它的 集合由消息的类型决定。STUN 头包含 STUN 消息类型，事务 ID 和长度。消息类型可以是 绑定请求(Binding Request)，绑定响应(Binding Response)，绑定错误响应(Binding Error Response)，共享私密请求(Shared Sercet Request)，共享私密响应(Shared Secret Response)，或共享私密错误响应(Shared Secret Error Response)。 事务 ID 用来关联请求和响应。长度指示 STUN 负载的总长度，不包括头。允许 STUN 运行 在 TCP 上。共享私密请求经常通过 TCP 发送（其实，是使用在 TCP 上的 TLS）。

STUN 协议定义了几个 STUN 属性。

1. MAPPED-ADDRESS 属性，这是 IP 地址和端口号。 它经常放在绑定响应中，并且表示服务器看见的绑定请求的源 IP 地址和端口号。
2. 是 RESPONSE-ADDRESS 属性，可以出现在绑定请求中，且表示绑定响应发送的目的地。它 是可选的，如果不存在，绑定响应应发送给绑定请求的源 IP 地址和端口号。
3. CHANGE-REQUEST 属性，且它包含两个标志来控制用来发送响应的 IP

地址 和端口号。这些标志称为“改变 IP”和“改变端口号”标志。CHANGE-REQUEST 属性只 允许在绑定请求中。“改变 IP”和“改变端口号”标志对于判断客户是否在限制锥形 NAT 或限制端口锥形 NAT 之后是有用的。它们命令服务器从不同的源 IP 地址和端口号发送绑定 响应。CHANGE-REQUEST 属性在绑定请求中是可选的。

4. CHANGED-ADDRESS 属性。它存在于绑定响应中。它表明，如果客户请求了“改变 IP”和“改变端口号”行为，客户应该使用的源 IP 地址和端口号。
5. SOURCE-ADDRESS 属性。它只存在于绑定响应中。它表明响应发送的源 IP 地址和端口号。用于检测两次 NAT 的配置。
6. USERNAME 属性。它存在于共享私密响应中。它为客户提供一个临时的用户名和密码（在 PASSWORD 属性中编码）。USERNAME 还存在于绑定请求中，用作共 享私密的索引，并用来对绑定请求进行完整性保护
7. PASSWORD，只存在于 共享私密响应消息中。
8. MESSAGE-INTEGRITY 属性。它包含检查绑定请求和 绑定响应的完整性的消息。
9. ERROR-CODE 属性。它存在于绑定错误响应和共享私密错误响应中。它指示所发生的错误。
10. UNKNOWN-ATTRIBUTES 属性，它存在于绑定错误响 应或共享私密错误响应中。它表明请求的命令属性未知。
11. REFLECTED-FROM 属性，存在于绑定响应中。表明绑定请求发送者的 IP 地址和端口号，用于阻止并跟踪某类 拒绝服务器攻击。

3.1.2.1 STUN 消息头

所有的 STUN 消息包含 20 字节的头[9]。

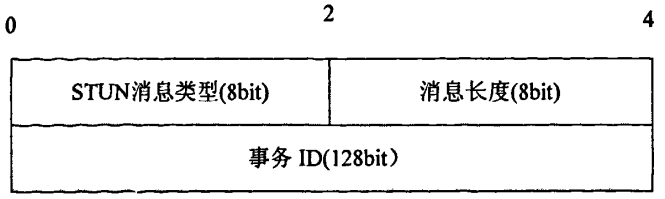


图 3-1 STUN 消息结构图

STUN 消息类型占用 2 字节，具体类型对应如下

0x0001：Binding Request

0x0101：Binding Response

0x0111：Binding Error Response

0x0002：Shared Sercet Request

0x0102：Shared Secret Response

0x0112：Shared Secret Error Response

消息长度占用 2 字节，表示除了消息头的这 20 字节外，剩余消息的长度

事务 ID 是一个长度为 128bit 的标识符（16 字节） 所有的响应消息和它所对应的请求消息的标识符是相同的。

3.1.2.2 STUN 属性

STUNT 消息头后面可以跟 0 个或者多个 STUN 属性[9]。每个消息包含一个两字节的属性类型和一个两字节的属性长度，以及一个大小不固定的属性值。

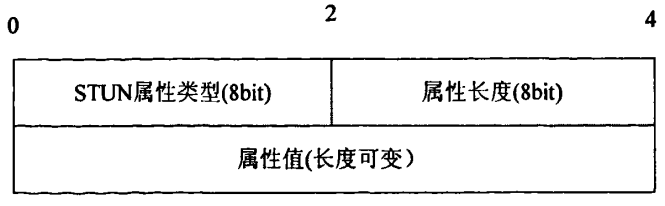


图 3-2 STUN 属性结构图

STUN 属性类型占用 2 字节，具体类型对应如下

0x0001：MAPPED-ADDRESS

0x0002：RESPONSE-ADDRESS

0x0003：CHANGE-REQUEST

0x0004：SOURCE-ADDRESS

0x0005：CHANGED-ADDRESS

0x0006：USERNAME

0x0007：PASSWORD

0x0008：MESSAGE-INTEGRITY

0x0009：ERROR-CODE

0x000a：UNKNOWN-ATTRIBUTES

0x000b：REFLECTED-FROM

STUNT 属性长度是表示属性值这段的字节数。而属性值的长度则依赖于属性类型以及这一节后面的定义。

为了允许将来对本规范进行修订时需要增加新的属性，将属性空间分为可选部分和强制部分。即使服务器或者客户端无法解析这种属性在请求和响应中它一样会被处理。 而小于 0x7fff 的属性则被认为是托管类型。除非它能被服务器或者客户端节解析 否则它将无法在请求和响应中被处理。消息中期待之外的属性将被抛弃。比如在一个 Binding Reques 消息包中收到一个 MESSAGE-INTEGRITY 的属性，哪么这个属性必须被忽略。

表 3-1 标明了哪一种属性属于哪一种消息类型。表中的字母 M 表示这个属性属于被托管类型(Mandatory)，字母 O 代表这个属性属于可选类型（Optional），字母 C 表示这个属性属于条件类型(Conditional)，根据消息语境的不同有不同表现。而 N/A 则表示属性不属于这个消息类型

表 3-1 STUN 消息属性关系表

属性名称	Binding Req.	Binding Resp.	Binding Error.	Shared Sercet Req.	Shared Sercet Resp.	Shared Sercet Error
MAPPED-ADDRESS	N/A	M	N/A	N/A	N/A	N/A

RESPONSE-ADDRESS	O	N/A	N/A	N/A	N/A	N/A
CHANGE-REQUEST	O	N/A	N/A	N/A	N/A	N/A
SOURCE-ADDRESS	N/A	M	N/A	N/A	N/A	N/A
CHANGED-ADDRESS	N/A	M	N/A	N/A	N/A	N/A
USERNAME	O	N/A	N/A	N/A	M	N/A
PASSWORD	N/A	N/A	N/A	N/A	M	N/A
MESSAGE-INTEGRITY	O	O	N/A	N/A	N/A	N/A
ERROR-CODE	N/A	N/A	M	N/A	N/A	M
UNKNOWN-ATTRIBUTES	N/A	N/A	C	N/A	N/A	C
REFLECTED-FROM	N/A	C	N/A	N/A	N/A	N/A

属性 MAPPED-ADDRESS

MAPPED-ADDRESS 属性表示映射过的 IP 地址和端口。它包括 8 位的地址族，16 位的 端口号，接着是长度固定的值，用来表示 IP 地址。 考虑到地址对齐前 8 位留空

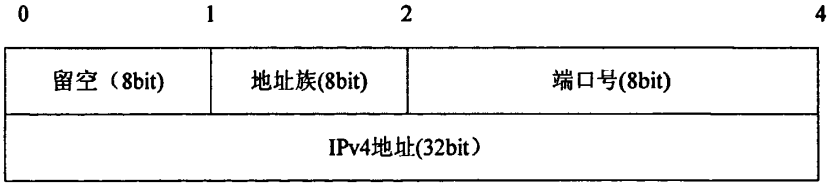


图 3-3 MAPPED-ADDRESS 属性结构图

端口号为网络字节序，表示映射的端口号。地址族通常是 0x01，与 IPv4 相对应。 MAPPED-ADDRESS 的前 8 位乎略，只是作为本地边界对齐参数。IPv4 地址是 32 位的。

属性 RESPONSE-ADDRESS

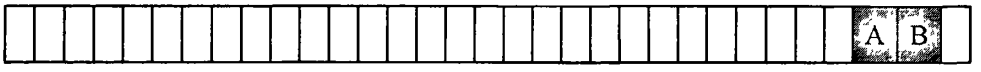
RESPONSE-ADDRESS 属性表示捆绑响应应该响应的目的地。它的语法与 MAPPED-ADDRESS 相同。

属性 CHANGED-ADDRESS

如果捆绑请求的 CNANGE-REQUEST 属性中的“改变 IP”和“改变端口”标志设置了， 则 CHANGED-ADDRESS 属性表示响应发出的 IP 地址和端口号。属性通常出现在捆绑响应 中，独立于标志的值。它的语法与 MAPPED-ADDRESS 相同。

属性 CHANGE-REQUEST

客户使 CHANGE-REQUEST 属性来请求服务器使用不同的地址和/或端口号来发送响应。属性是 32 位长的，但是只用 2 位（A-30 和 B-31）：



标志意义是：

A：这是“改变 IP”标志。如果为真，它请求服务器将捆绑响应发送到不同的 IP 地址， 而不是收到捆绑请求的地址。

B：这是“改变端口”标导。如果为真，它请求服务器将捆绑响应发送到不同的端口， 而不是收到捆绑请求的端口。

属性 SOURCE-ADDRESS

SOURCE-ADDRESS 属性出现在捆绑响应中。它表示服务器发送响应的源 IP 地址和端 口。它的语法与 MAPPED-ADDRESS 相同。

属性 USERNAME

USERNAME 属性用于消息的完整性检查。它的意思是在消息完整性检查中标识共享私密。USERNAME 通常出现在共享私密响应中，与 PASSWORD 一起。当使消息完整性检查 时，可有选择地出现在捆绑请求中。

USERNAME 的值是变长且不透明的。它的长度必须是 4 的倍数（以字节为单位），以保证属性与字边界对齐。

属性 PASSWORD

PASSWORD 属性用在共享私密响应中。它通常出现在共享私密响应中，与 USERNAME 一起。

PASSWORD 的值是变长的，用作共享私密。它的长度必须是 4 的倍数（以

字节为单位），以保证属性与字边界对齐。

属性 MESSAGE-INTEGRITY

MESSAGE-INTEGRITY 属性包含 STUN 消息的 HMAC-SHA1。它可以出现在捆绑 请求或捆绑响应中。由于使用 SHA1 散列算法，HMAC 将会是 20 个字节。用作 HMAC 输 入的文本是 STUN 消息，包括头部，直到且包括 MESSAGE-INTEGRITY 属性前面的属性。文本填充 0 以使其 64 字节对齐。 其结果是， MESSAGE-INTEGRITY 属必须是任何 STUN 消息的最后一个属性。用作 HMAC 输入的键值取决于其内容。

属性 ERROR-CODE

ERROR-CODE 属性出现在捆绑错误响应或共享私密错误响应中。它的数值范围从 100 到 699，加上以 UTF-8 编码的文本原因语句，且固定于它的代码分配，SIP 的语义和 HTTP。原因语句用于用户提示，且可以是表示原因代码的任何适当的语句。原因语句的 长度必须是 4 的倍数（以字节为单位）。如果需要，可以通过在文本的末尾加入空格来完成。所定义的响应号的缺省原因语句表述如下。

为有助于处理，从余下的代码中对错误代码（百位数）的分类进行单独编码。

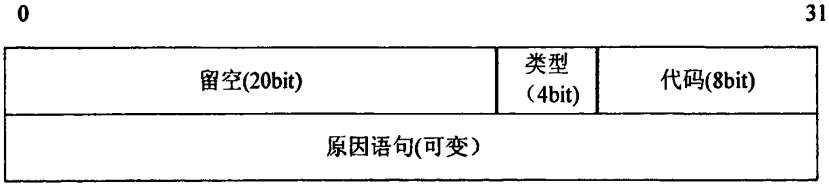


图 3-4 属性 ERROR-CODE 结构图

分类表示响应号的百位数。其值必须在 1 至 6 之间。数字表示响应号的 100 的模数，其值必须在 0 至 99 之间。

下面的响应号，与它们缺省的原因语句（括号中）一起，目前定义为：

400 (Bad Request)：请求损坏。客户在修改先前的尝试前不应该重试该请求。

401 (Unauthorized)：捆绑请求没有包含 MESSAGE-INTEGRITY 属性。

420 (UnknownAttribute)：服务器不认识请求中的强制属性。

430 (Stale Credentials)：捆绑请求没有包含 MESSAGE-INTEGRITY 属性，但它使用过期 的共享私密。客户应该获得新的共享私密并再次重试。

431 (Integrity Check Failure)：捆绑请求包含 MESSAGE-INTEGRITY 属性，但 HMAC 验证失败。这可能是潜在攻击的表现，或者客户端实现错误。

432 (Missing Username)：捆绑请求包含 MESSAGE-INTEGERITY 属性，但没有 USERNAME 属性。完整性检查中两项都必须存在。

433 (Use TLS)：共享私密请求已经通过 TLS 发送，但没有在 TLS 上收到。

500 (Server Error)：服务器遇到临时错误。客户应该再次尝试。

600 (Global Failure)：服务器拒绝完成请求。客户不应该重试。

属性 UNKNOWN-ATTRIBUTES

UNKNOWN-ATTRIBUTES 属性只存在于其 ERROR-CODE 属性中的响应号为 420 的捆绑错误响应或共享私密错误响应中。

属性包含 16 位值的表，每个表示服务器不认识的属性类型。如果未知属性数量是奇数， 则表中的属性之一必须重复，以使表的总长度是 4 字节的倍数。

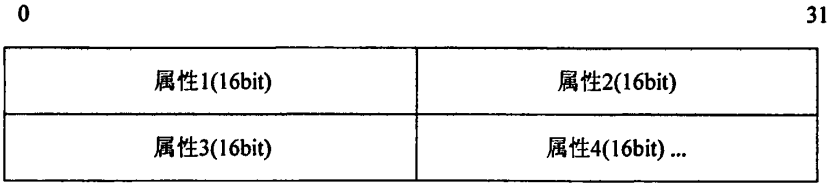


图 3-5 UNKNOWN-ATTRIBUTES 属性结构图

属性 REFLECTED-FROM

REFLECTED-FROM 属性只存在于其对应的捆绑请求包含 RESPONSE-ADDRESS 属性 的捆绑响应中。属性包含请求发出的源的身份（按照 IP 地址）。它的目的是提供跟踪能力， 这样 STUN 就不能被用作 DOS 攻击的反射器。其语法与 MAPPED-ADDRESS 属性相同。

3.1.3 STUN 工作原理

3.1.3.1 网络状态检测

首先定义两个实体：

STUN Client: STUN 客户端是一个能够产生 STUN 请求的实体，它能够嵌入在终端系统的应用程序中，一般在局域网中运行。

STUN Server:STUN 服务器是一个接手 STUN 请求的实体并且能够产生 STUN 响应,STUN 服务器一般是在公网上运行，它是无状态的。

首先 STUN 协议需要检测网络状态[9]:

STUN 协议中所定义的网络状态有：公网，有防火墙阻塞，完全锥体(Full cone) NAT 限制锥体(Restricted cone) NAT，端口限制锥体(Port restricted cone)NAT，对称性(Symmetric) NAT 几种。

检测过程如图 3-6 所示

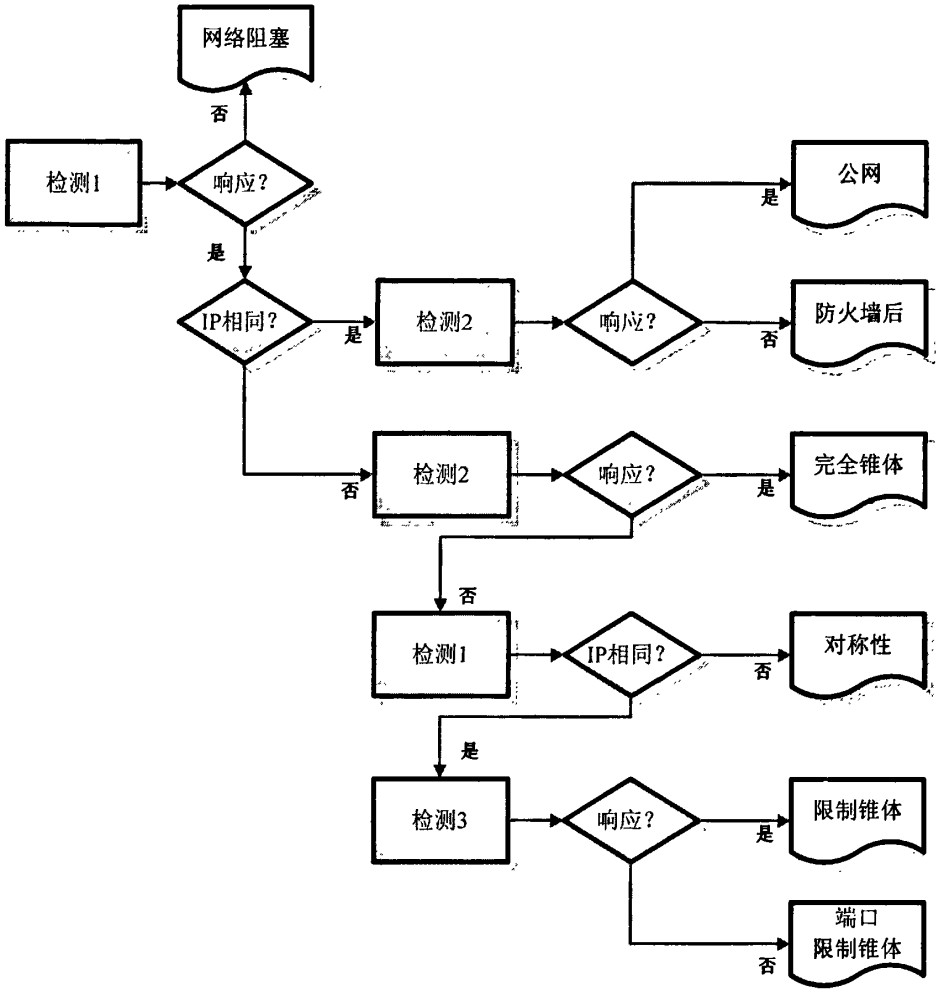


图 3-6 STUN-网络状态检测流程图

检测 1，客户发送 STUN Binding Request 给服务器。不设置任何 CHANGE-REQUEST 的属性，也没有 RESPONSE-ADDRESS。

属性。服务器接收到该请求后会用相同的地址，端口发送回复消息。

检测 2，客户发送了含有设置了 Change Ip 以及 Change Port 属性的请求。

检测 3，客户发送了含有设置了 Change Port 属性的请求。

（1）STUN 客户端以检测 1 开始向 STUN 服务器发送出 STUN 请求，如果这个测试没有产生响应，客户端会马上知道它没有 UDP 连接的能力。如果这个测试产生一个响应，那么客户端将检查 MAPPED-ADDRESS,如果这个地址和本地地址相同。

并且端口与发送这个 Socket 的端口也相同，哪么这个客户端知道它没有被 NAT,于是执行步骤 2，否则执行步骤 5。

（2）STUN 客户端发送一个带有 Change Ip 以及 Change Port 属性的请求开始检测 2

（3）STUN 服务器收到 STUN 客户端发来的带有 Change Ip 以及 Change Port 属性的请求后，Change Ip 以及 Change Port 要求以不同的地址和端口发送响应，于是 STUN 服务器使用另一个端口，IP 地址发送响应，或者向其他 STUN 服务器发送一个带有 RESPONSE 属性的 STUN 请求，请求另一个服务器接收 STUN 客户端请求时所得到的关于这个请求的源地址。

（4）如何 STUN 客户端收到了这个响应说明它是在公网上可到达的，否则是在 UDP 防火墙后，测试结束。

（5）STUN 客户端发送一个带有 Change Ip 以及 Change Port 属性的请求开始测试 2。

（6）同步骤 3，对于这个请求由不同 IP 地址，端口的服务器来响应。

（7）如果 STUN 客户端收到了响应，则表示它在完全锥体(Full cone) NAT 后，测试结束，否则进行步骤 9。

（8）STUN 客户端进行检测 1 的一个变动版，检测 1 本来是用来向 STUN 服务器发送请求的。而这个变动版则向 STUN 服务器发送一个不同 IP 地址的请求, 这个请求与测试 1 类似, 在配置中 STUN 客户端只知道 STUN 服务器的地址。STUN 在步骤 1 的响应中已经从 CHANGED-ADDRESS 属性得到了另一个 Server 的地址和端口。STUN 客户端将从这个服务器收到的响应信息中的 MAPPED-ADDRESS 的值与在步骤 1 中得到响应消息中的 MAPPED-ADDRESS 相比较，如果不同，则它处于对称性(Symmetric) NAT 后，测试结束，否则进入步骤 9。

(9) STUN 客户端向 STUN 服务器发送一个仅设置有 Change Port 标识的请求，进行检测 3，这时 STUN 服务器以不同的端口向 STUN 客户端发送响应。如果 STUN 客户端在收到了这个响应，则它处于 之后，否则处于 之后，测试结束。

3.1.3.2 NAT 穿越流程

STUN是采用打洞（Hole Punching)的方式来实现UDP的NAT穿越的[10]。 原理如图标3-7所示。

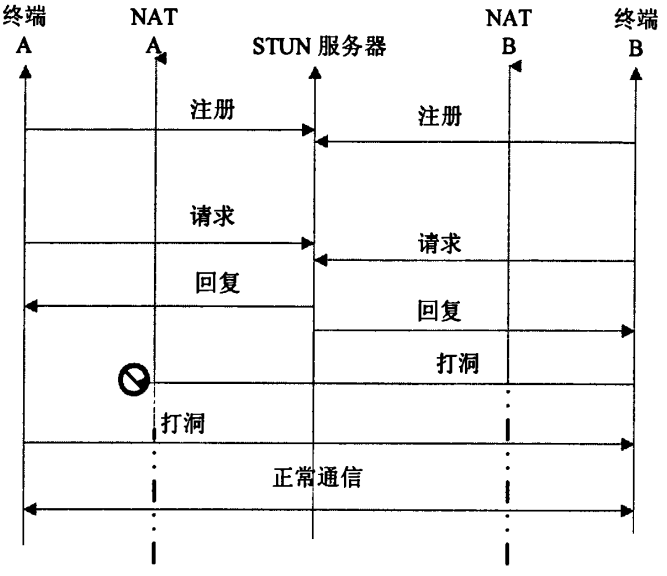


图 3-7 STUN 工作流程图

要穿越NAT进行通信，首先要知道对方在其NAT上对应的外网端口号，这需要一台拥有公网地址的STUN服务器，NAT后的终端向STUN服务器发送消息，终端发出的IP包通过NAT的转换后发送往服务器，服务器接收到这个包后就可以解析出其中的地址信息，并记录下来。这一过程我们称之为注册

假设终端A和终端B需要通讯 他们各发送连接请求到STUN服务器，服务器会把对方所映射的公网地址告诉双方。这样终端A,B就知道对方所映射在公网上的地址了。

接着通知终端B发送一个数据包到终端A,由于终端A前NAT A的存在，这个消息包是无法到达终端A的，但对于终端B前的NAT B会认为此次会话的发起方为终端B,从此不会阻挡从终端发送给终端B的消息包，我们称这在NAT B上打了一个洞。之后服务器通知终端A发送一个数据包到终端B,由于之前打洞的存在这个消息可以穿越NAT B,到达终端B,同时在NAT A上打了一个洞。此后就可以脱离服务器进行正常的通信了。NAT穿越至此完成。

3.2 基于 TCP 的 NAT 穿越

3.2.1 现有 TCP 穿越 NAT 技术的比较

在目前流行的 TCP 穿越技术中，STUNT, NATBlaster, Peer-to-Peer NAT 是最被看好的。

3.2.1.1 STUNT

NUTSS 是康乃尔大学提出一个穿透不同 NAT 直接连接 TCP 的项目，NUTSS 的目的是达到全球互通的境界，让所有在不同架构下的节点彼此之间都能够建立联机，不会因为在 NAT 后而受到影响[11]。NUTSS 项目组为此开发了 STUNT（Simple Traversal of UDP through Nats and TCP）协议去开启 NAT 后节点的 TCP 连接。，两个不同 NAT 后的节点在建立联机的初期，先透过第三者（STUNT 里称之为 STUNT 服务器）协助建立联机，联机建立完成后，两个节点便可直接做数据的传输，不需再经过第三者转传。 STUNT 是通过低 TTL 的 SYN 信号来获取终端在 NAT 所映射的 IP 以及端口号，然后利用发送 SYNACK 欺骗（模型#1 见后文） 或者打洞（模型#2 见后文）来实现 NAT 穿越的[12]。

3.2.1.2 NATBlaster

NATBlaster 和 STUNT 模型#1 类似但是在发送伪造的 SYNACK 包这一部分有所不同[13]。开始时终端发出一个低 TTL 的 SYN，并记住协议栈所使用的 TCP 序号。与前面的类似，SYN 包在网络中被丢弃。两台主机之间交换各自的序号，这时不是由 STUNT 服务来产生伪造 SYNACK 包二是由每台主机各自手工产生一个对方期望收到的 SYNACK 包。手工产生的包通过 RAW socket 注入网络。所以这和 STUNT 1#欺骗不同，因为该包中的源地址与注入该包的终端地址匹配。一旦收到 SYNACK，ACK 就被交换了，从而完成了连接建立。

3.2.1.3 Peer to Peer NAT

P2PNAT 方法利用的是 TCP 协议中所定义的同时打开（Simultaneous Open）[14]，两个终端都通过发送 SYN 包来创建一个连接[15]。如果 SYN 包在网络中交叉（穿过），这两个终端协议栈都用 SYNACK 包应答从而建立连接。如果在对方的 SYN 包离开它的 NAT 之前，一个终端的 SYN 包到达该终端的 NAT 而被丢

弃，那么，第一个终端的协议栈会在 TCP 同时打开之后而关闭，而另一个终端则会按照正常流程打开。接下来的情况，链路中的包看起来像 STUNT 模型 2#，这种方法中不需要用到低 TTL 的 SYN 消息，也不需要相关的 ICMP。但是该方法需要 NAT 允许在一个向外发送的 SYN 之后紧跟一个接收一个向内发送的 SYN，而且，该方法需要终端在一个循环中不停的尝试重连，直到超时。如果 SYN 包并没有被 NAT 所丢弃，也没有返回一个 TCP 连接错误，那这种方法将陷入包泛滥的境地，直到超时。

3.2.1.4 三种技术之间的比较

表 3-2 TCP 穿越技术比较表

	低 TTL	同时打开	超级权限
STUNT #1	需要	不需要	需要
STUNT #2	需要	不需要	不需要
NATBlaster	需要	不需要	需要
Peer to Peer NAT	不需要	需要	不需要

如表 3-2 其中 STUNT #1 和 NATBlaster 需要获得超级权限才能进行 SYNACK 欺骗，所以应用性受限制，而 Peer to Peer NAT 需要终端支持 TCP 协议中“同时打开”这个性质。虽然 Linux 可以完美支持这一功能。而 Windows 操作系统在 XP sp2 之前对此都不支持。考虑到 VNC 的使用者并不一定都拥有超级权限，也不一定都装有 Windows XP sp2 以上的操作系统版本。

所以在这一方案中我们选择了 STUNT 作为我们的 TCP 穿越方案。

3.2.2 STUNT 消息结构

STUNT 协议中的消息，是使用 Big-endian 编码的 TLV(type-length-vlaue)字节流[16]。所有的 STUNT 消息都有一个 STUNT 消息头，和一个或者数个属性。而属性的具体内容则依赖于消息类型以及属性的类型。消息头将包含消息类型，事务 ID,以及消息长度。消息类型可以分为 Binding Request, Binding Response, Shared Secret Request, Shared Secter Response, Capture Request, Capture Acknowledgement or Capture Response. 而事务 ID 则用来关联请求和响应。

STUNT 在 STUN 的基础上定义了自己的几种属性类型。第一种是

PACKET-REQUEST 属性， PACKET-REQUEST 属性包含着一些包类型，包括 TCP 协议包类似 SYN,SYNACK etc, ICMP 包或者用户自定义类型的包。它同时也包含一些象征性的标志比如这个包应该被发送到服务端，还是应该被发送到客户端，这个包是从某个服务器地址发出的还是要发送到这个服务器地址，或者这个包的重发频率为多少之类。一个 Capture Request 消息将包含一个或者多个 PACKET-REQUES 属性。

第二种属性是 CAPTURE-TIMEOUTS 定义了捕捉测试的周期。它还包含了指定包的转发间隔等信息。

第三种属性是 CAPTURED-PACKET 它包含了 IP 包头和包是从服务器发出还是发往服务器的信息。它还包含了这个包是什么时候被第一次发送的和它被转发过多少次这样的信息。

3.2.2.1 STUNT 消息头

和 STUN 类似，所有的 STUNT 消息头固定为 20 字节[16]。

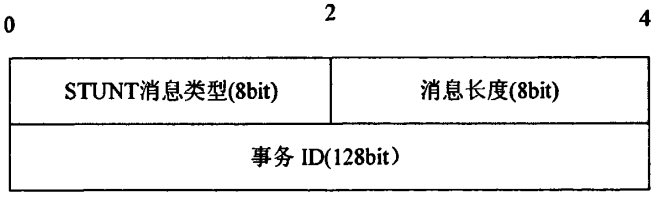


图 3-8 STUNT 消息结构图

STUNT 消息类型占用 2 字节，新增的具体类型对应如下

0x0003: Capture Request

0x0103: Capture Response

0x0113: Capture Error Response

0x0123: Capture Acknowledge Response

消息长度占用 2 字节，表示除了消息头的这 20 字节外，剩余消息的长度

事务 ID 是一个长度为 128bit 的标识符（16 字节） 所有的响应消息和它所

对应的请求消息的标识符是相同的。

3.2.2.2 STUNT 属性

STUNT 消息头后面可以跟 0 个或者多个 STUNT 属性.每个消息包含一个两字节的属性类型和一个两字节的属性长度，以及一个大小不固定的属性值[16]。

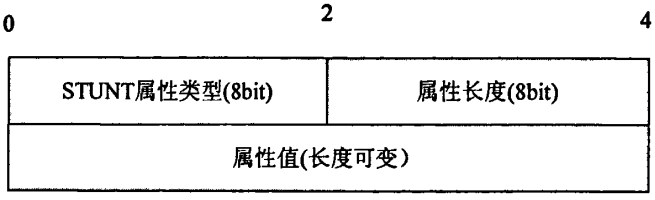


图 3-9 STUNT 属性结构图

STUNT 属性类型占用 2 字节，具体类型对应如下

0x0100: PACKET-REQUEST

0x0101: CAPTURE-TIMEOUTS

0x0102: CAPTURED-PACKET

STUNT 属性长度是表示属性值这段的字节数。而属性值的长度则依赖于属性类型以及这一节后面的定义。

无论是是属性类型大于 0x7fff 的属性被认为是可选的，即使服务器或者客户端无法解析这种属性在请求和响应中它一样会被处理。 而小于 0x7fff 的属性则被认为是托管类型。除非它能被服务器或者客户端节解析 否则它将无法在请求和响应中被处理。消息中期待之外的属性将被抛弃。比如在一个 Capture Request 消息包中收到一个 CAPTURED-PACKET 的属性，哪么这个属性必须被忽略。

表 3-3 标明了哪一种属性属于哪一种消息类型。表中的字母 M 表示这个属性属于被托管类型(Mandatory), 字母 O 代表这个属性属于可选类型（Optional），字母 C 表示这个属性属于条件类型(Conditional)，根据消息语境的不同有不同表现。而 N/A 则表示属性不属于这个消息类型

表 3-3 STUNT 消息属性对应表

属性	Capture Req.	Capture Ack.	Capture Resp.	Capture Error
PACKET-REQUEST	M	N/A	N/A	N/A
CAPTURE-TIMEOUTS	M	N/A	N/A	N/A
CAPTURED-PACKET	N/A	N/A	C	N/A
CHANGED-ADDRESS	N/A	M	N/A	N/A
RESPONSE-ADDRESS	C	N/A	N/A	N/A
ERROR-CODE	N/A	N/A	N/A	M
UNKNOWN-ATTRIBUTES	N/A	N/A	N/A	C
SERVER	N/A	O	O	O
MESSAGE-INTEGRITY	O	O	O	N/A

属性：PACKET-REQUEST

这个属性被用来向服务器发送一个特定类型的数据包，或者等待一个特定类型的数据包的到达。作为这一属性的作用之一，这一属性可以被用来通过 TCP 条件状态机来管理一套事务。这个属性的属性值结构如下

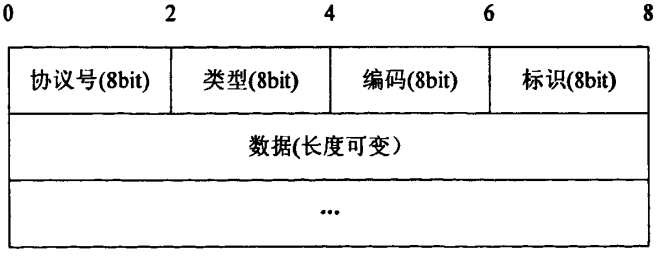


图 3-10 PACKET-REQUEST 属性结构图

协议号为 IP 协议的代码，如果协议号为 0x06(TCP).那么这个属性为 TCP 类型。标识位和编码位将为 0。假如协议号为 0x01(ICMP)，那么这个属性为 ICMP 类型。类型位上所表示的是 ICMP 类型，而编码位则为 ICP 编码。当协议号为 TCP 和 ICMP 的时候 Data 位将为空。

新增的两种协议类型定义如下:

0x00: 这个包将不能被发送或者和任何一个接收到的消息包成功配对。这个包必须被用于通知服务器等待下一个捕获动作超时的发生。

0xFF: 这个包是为客户端服务的。数据必须包含一个有效的 IP 包。这个 IP 包是以零作为填充向前对齐的。IP 包的总长度字段将会在服务端被抛弃，而只能用于客户端。

PACKET-REQUEST 标识位的属性含义如下

A 这是一个发送或者接收标识。如果它为真，那么服务器应该发送这个特定的包出去；如果它为假，哪么服务器将会等待一个特定的包的到来。

B 这是一个 ACK 收到但是依然转发标识。只对含有标识 A 的 TCP 数据包有效。当它为真的时候，服务器应该继续转发这个包，即使服务器已经收到了它的 TCP ACK 回执。

C 这是一个延迟标识。只对含有标识 A 的数据包有效。当它为真的时候，这个包将会被延迟，而延迟的时间由 CAPTURE-TIMEOUTS 属性决定。

D 这是一个 IP 改变标识。当它为真的时候，这个包将以一个不同的 IP 地址发送。这个不同的 IP 地址将通过 Response 的回执中的 CHANGED-ADDRESS 属性发送到客户端。

E 这是一个 Port 改变标识。当它为真的时候，这个包将以一个不同的端口发送。这个标识只对 TCP 数据包有效。这个不同的端口将通过 Response 的回执中的 CHANGED-ADDRESS 属性发送到客户端。

属性: CAPTURE-TIMEOUTS

Capture Timeout，是服务器应该发送或者接收一个包所需要的总时间，这个时间是以微秒为单位表示的。Delay Timeout，是含有上节提到的标识 C 的包应该被延迟多久，这个时间是以微秒为单位表示的。另外含有 C 标识的包的转发延迟时间将是 Delay Timeout 的两倍。

属性：CAPTURED-PACKET

类型 0x00 表示这个包是被服务器发出的，而类型 0x01 表示这个包是被发往服务器的。计数是用来表示与这个包同样的拷贝被发送或者接收过几次。包时间用来表示从收到第一个拷贝后经过的时间，以秒为单位表示。包数据包含着以零对齐的 IP 数据包。

3.2.3 STUNT 工作原理

3.2.3.1 网络状态检测

与 STUN 不同，STUNT 还需要检测 NAT 映射类型和终端过滤类型[16]。检测 NAT 的映射方式也同样用了三种检测:

检测 1,客户使用一个 socket 绑定到本地的某个地址，如 59.65.128.1:100,然后连接服务器地址如 59.65.128.100:5050。连接成功后发送绑定请求，服务器返回 TCP 连接 NAT 后的合法地址。

检测 2，客户使用一个 socket 绑定到和检测 1 同样的本地地址，如 59.65.128.1:100,然后连接服务器地址但是端口和测试 1 所连接的不同如 59.65.128.100：5060。这个地址可以从检测 1 中的服务器响应中得到。连接成功后，发送绑定请求，服务器返回响应。

检测 3，和上述两个测试相似，但是连接的是不同的服务器地址如 59.65.128.200：5050。这个地址可以从检测 1 中的服务器响应中得到。连接成功后，发送绑定请求，服务器返回响应。

检测过程如图 3-11 所示

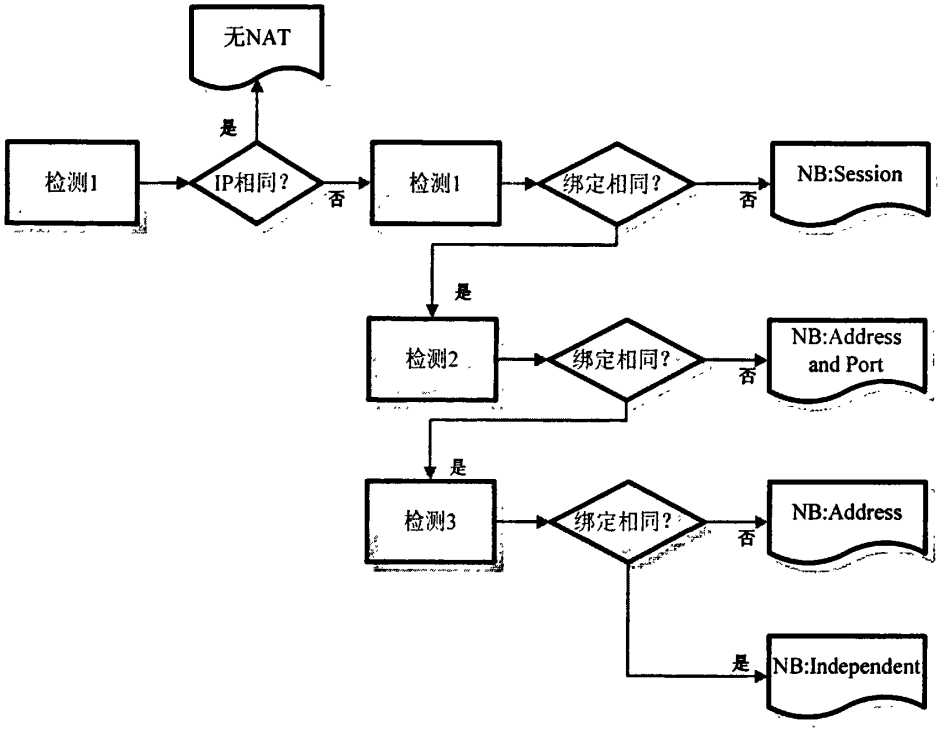


图 3-11 NAT 映射类型流程图

（1）执行检测 1，得到服务返回的 MAPPED-ADDRESS，并与本地地址比较，如果一致，说明客户不在 NAT 后，检测结束。

（2）关闭检测 1 的连接后，再次执行检测 1，得到服务器返回的 MAPPED-ADDRESS，并与第（1）步获得的 MAPPED-ADDRESS 进行比较，如果比较结果不一致，表示 NAT 的类型是连接依赖的 NB:Session。 检测结束

（3）执行检测 2，得到服务器返回的 MAPPED-ADDRESS，并与第（1）步通过检测 1 所得到的 MAPPED-ADDRESS 进行比较，如果不相同则表明 NAT 类型是地址端口依赖的 NB:Address and Port。

（4）最后执行检测 3，得到的 MAPPED-ADDRESS 也和第（1）步中通过检测 1 中的 MAPPED-ADDRESS 进行比较们如果不同那么 NAT 的映射方式为 NB:Address,相同则是 NB:Independent。

3.2.3.2 NAT 穿越流程

两个不同 NAT 后的节点在建立联机的初期，先透过第三者（STUNT 里称之为 STUNT 服务器）协助建立联机，联机建立完成后，两个节点便可直接做数据的传输，不需再经过第三者转传。 STUNT 是通过低 TTL 的 SYN 信号来获取终端在 NAT 所映射的 IP 以及端口号，然后利用发送 SYNACK 欺骗（模型#1） 或者打洞（模型#2）来实现 NAT 穿越的[17]。

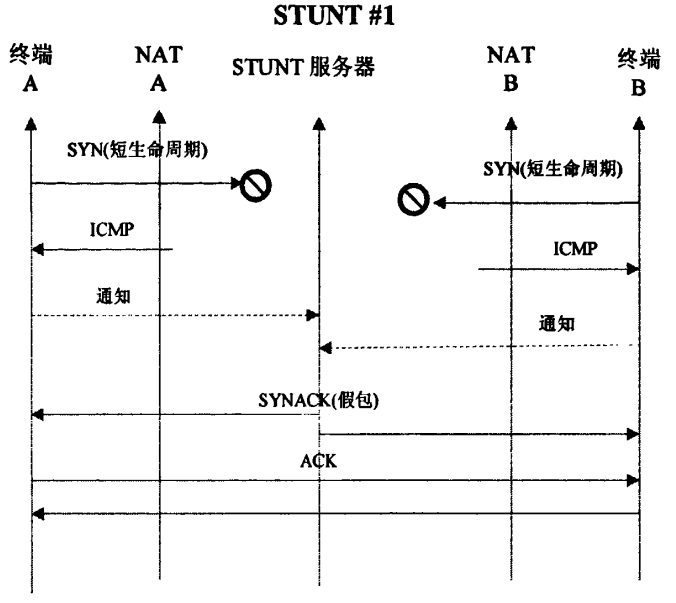


图 3-12STUNT 模型 1 工作原理图

如图标 3-12 所示，终端双方都主动发送低生命周期（TTL）的 SYN，且 SYN 的生命周期要足够大，大到 SYN 包能穿过他们各自的 NAT，但生命周期又要足够小，小到穿过各自的 NAT 后就过期而被网络丢弃。通过侦听经过 RAW-socket 或 PCAP 返回的 ICMP 消息[19]，分析返回的 ICMP 消息，终端可以得知初始 TCP 的序号。终端双方把各自的序号告诉都能抵达（连接）的 STUNT 服务器，紧接着 STUNT 服务器通过设置匹配的序号构造一个 SYNACK 来欺骗双方（双方都认为该 SYNACK 是对方回应的）。完成 TCP 握手任务的 ACK 按照正常方式穿过网络。这种方法最大的缺点在于，它需要一个第三方为一个任意地址产生一个欺

骗包，而该欺骗包很可能被网络中各种各样的进出过滤器或者防火墙丢弃或者拦截。

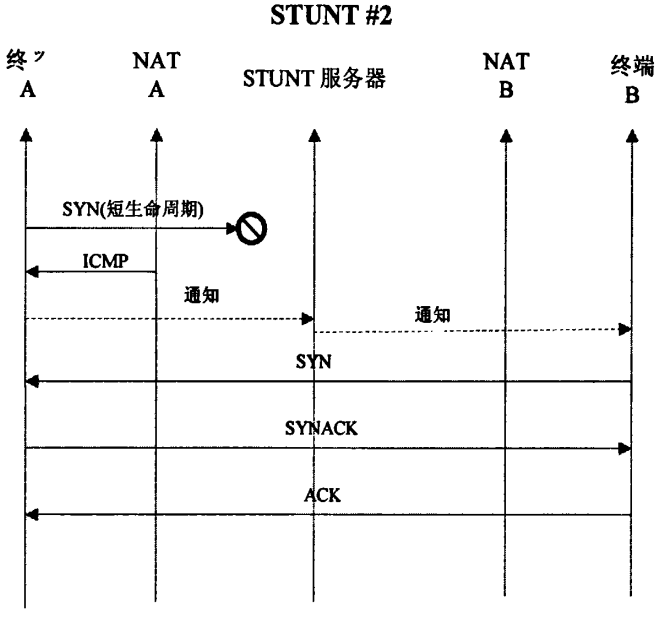


图 3-13 STUNT 模型 2 工作原理图

如图3-13所示，终端A 发出一个低生命周期(TTL)的SYN包，同样这个SYN的生命周期要足够大，大到SYN包能穿过他们各自的NAT，但生命周期又要足够小，小到穿过各自的 NAT 后就过期而被网络丢弃。终端 A 通过侦听经过RAW-socket 或 PCAP 返回的 ICMP 消息，分析 ICMP 消息可以得知初始 TCP 的序号。终端 A 把自己的序号告诉的 STUNT 服务器，接着终端 A 取消该连接企图，并在相同的地址和端口创建一个被动的 TCP 套接字。然后 STUNT 服务器会通知另一个终端初始化一个正常的 TCP 连接。终端 B 经过终端 A 第一次尝试连接留下的洞，进行三次握手最终和终端 A 建立 TCP 连接。

3.2.4 提高 STUNT 的穿越性能

3.2.4.1 STUNT 所存在的问题

利用 STUN 实现 UDP 穿越已经相当成熟，但是基于 STUNT 实现的 TCP 穿

越方法却依然有有几个潜在的问题：第一，它需要终端确定一个 TTL，该 TTL 足够大，大到可以穿透自己的 NAT，但又要足够小，小到不能到达对方的 NAT；如果 TTL 不准确则会导致穿越失败，而这个 TTL 和网络的情况是密切相关的，目前已知的 TTL 预测算法都不能令人满意，而且在 Window 操作系统下，对 TTL 的设置还存在着严重的问题，即实际 TTL 和设定 TTL 经常有很大差距。第二，作为 SYN 包的回应，比 TTL 优先的 ICMP 错误是可能产生的，这种 ICMP 消息会被 NAT 认为是致命的错误而拦截掉；第三，NAT 可能改变初始 SYN 的 TCP 序号，这样就可能导致基于原始序号的欺骗 SYNACK 到达 NAT 时，被当作窗口之外的包；第四，如果采用模型#1，它需要一个第三方为一个任意地址产生一个欺骗包，而该欺骗包很可能被网络中各种各样的进出过滤器或者防火墙丢弃或者拦截。

以上几点都可能导致 STUNT 穿越失败 TCP 连接无法建立。影响了穿越的成功率。

3.2.4.2 不使用低 TTL 信号

在进行 STUNT 穿越试验的时候我们注意到在 Windows 下设置 TTL 生命周期有的时候会发生一些问题。发出的 SYN 信号并未根据设定的 TTL 生命周期衰减。而更近似于一个随机的数值，这和 Linux 下的反应完全不同。在 Windows 下 SYN 信号的生命周期经常会长到足以顺利到达对方 NAT。以 STUNT 模型 2#为例见图 4-1。

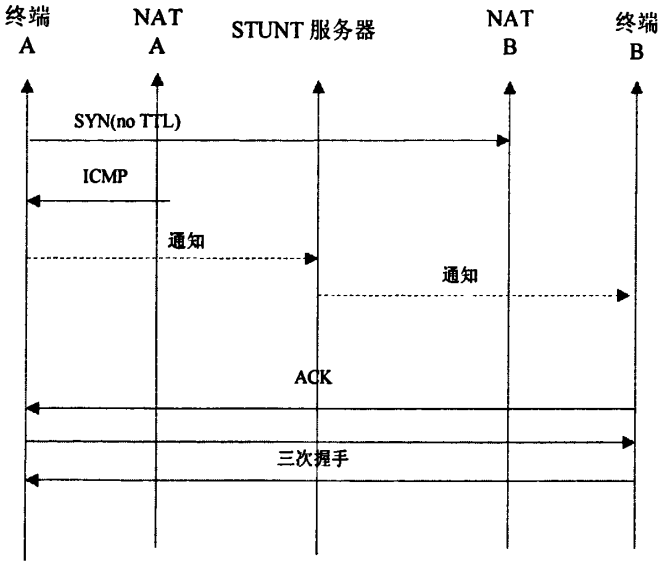


图 3-14 不使用低 TTL 信号图

如图 3-14 所示当来自终端 A 的 SYN 包被 NAT 过滤后，我们期待 NAT B 会返回一个 RST 错误通知终端 A 。但经过对北航实验室和石景山区政府所用 NAT 设备实验测试。发现大部分的 NAT 并没有这样作，见表 3-3，96%以上的 NAT 只是将这个 SYN 包安静的丢弃了，并没有做出任何响应，只有两台 NAT 返回了 RST 错误信息。而当 SYN 包被 NAT B 抛弃后，对于 NAT A 来说它会认为 SYN 没有到达 NAT B 就被消灭了，这和 STUNT # 2 模型中 SYN 因为生命周期结束而在到达 NAT B 之间消灭的情况实际是一模一样的。NAT A 同样会返回一个带有连接信息的 ICMP 消息来通知终端 A. 终端 A 既可以得到自己在 NAT A 所映射的 TCP 连接号等信息。 之后的流程就等同与 STUNT 模型#2 了，终端 A 终止连接并在相同的端口上建立一个监听，随后终端 B 向终端 A 发起三次握手成功建立连接，并传送数据。

表 3-3 北航实验室和石景山区政府所用 NAT 设备实验测试数据

设备厂商	型号	台数	丢弃数	比率
TP-Link	TL-R402M	4	4	100%
D-Link	DI-504M	5	5	100%
CISCO	WS-CE500-24TT	12	12	100%
3Com	3CFSU08	3	3	100%
LINKSYS	SD208	3	3	100%

杂牌		31	29	93%
总计		58	56	96.5%

那么就是说即使一开始发送的 SYN 信号不带有任何 TTL 信息。 STUNT#2 依然有很大的几率（96%以上）可以正常工作。而这种方法不带有 TTL 信息，可以避免由于 TTL 周期过长 或者过短造成的失败率,以及和 ICMP 早于 TTL 返回被 NAT 认为是致命错误这两个问题。由于 TTL 预测的实际成功率很有限，放弃 TTL 使用不带 TTL 信息的 SYN 在实际应用中反而可以获得更好的连接成功率，具体的例证见第四章。表 3-4 说明了使用 TTL 和不使用 TTL 之间的差异。

表 3-4 使用 TTL 和不使用 TTL 之间的比较

	预测 TTL	ICMP 错误	RST 错误
STUNT#2	是	是	否
STUNT#2 no TTL	否	否	是

对于不返回 RST 错误的 NAT 来说，显然不使用 TTL 信号将会比原方案的成功率有不少的提高。而对于少数返回 RST 错误的 NAT 来说即使使用低 TTL 信号也依然存在上述 TTL 周期过长 或者过短造成的失败率,以及和 ICMP 早于 TTL 返回被 NAT 认为是致命错误这两个问题不一定会连接成功。经过实验，我们发现相比之下不带 TTL 信息的 STUNT 模型反而比带 TTL 信息的 STUNT 有更高的成功率。

3.2.4.3 引入竞争

考虑不使用低 TTL 的 SYN 的 STUNT #2 模型 ，根据上一小节的分析我们可以看到由于请求是由通信的其中一方开始的，那么穿越是否成功依赖于 NAT B 在接收到一个应该被过滤掉的 SYN 包后是返回一个 RST 错误信息还是简单的将这个包丢弃。如果不满足这个条件那么很可能无法建立连接。

这种连接错误有强烈的方向性，就是说穿越的成功与否和发起方 NAT 的类型以及接收方的 NAT 类型有着直接关系。那么我们来看下面这种情况。假设 NAT A 不会返回一 RST 错误。而 NAT B 会返回一个 RST 错误，那么如果发起这次连接的终端位于 NAT A 后面的私网中，那么这次连接将会失败，反之如果发起这次连接的终端位于 NAT B 后面的私网中，这次连接则会成功。

为了消除这种方向性，我们决定在方案中引入竞争。即由终端 A,B 同时开始尝试发起连接，由服务器判断，先成功获得 ICMP 信息的一方为胜者。

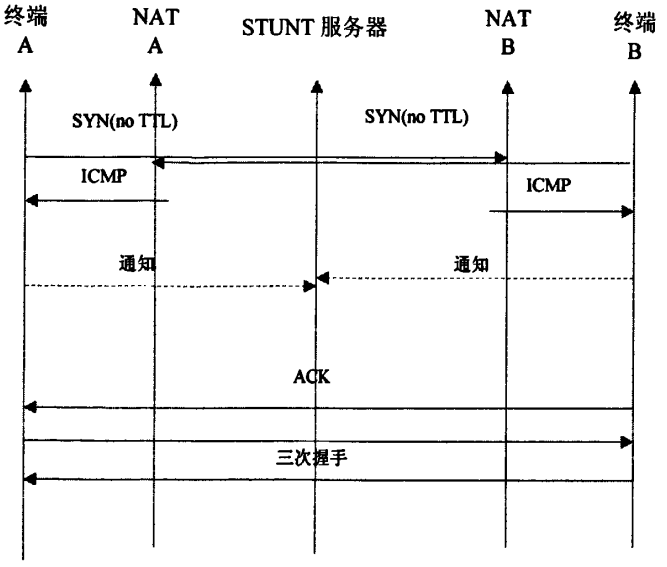


图 3-15 引入竞争后的流程图

如图 3-15 所示，基本流程和 STUNT #2 模型类似，只不过由终端 A,和终端 B 同时发起连接请求，两个终端分别通过侦听经过 RAW 套接字或 PCAP 返回的 ICMP 消息，得到初始 TCP 的序号后，各自把自己的序号告诉的 STUNT 服务器，接着服务器会通知先到达的一方比如说终端 A 取消该连接企图，并在相同的地址和端口创建一个被动的 TCP 套接字。然后 STUNT 服务器会通知另一个终端比如说终端 B 初始化一个正常的 TCP 连接。接下来的步骤和 STUNT#2 模型一样，进行正常的 TCP 三次握手建立连接，并发送数据。

和引入竞争之前的方案来比，引入竞争消除了原有的方向性差异。使得通信双方只要有任意一方不会返回 RST 错误就可以顺利建立连接。进一步提高了穿越的成功率。

3.2.4.4 与原方案的对比

改进方案主要涉及了两个方面，其中第一点不适用低 TTL 信号和第二点引入竞争。

改进方案中用无 TTL 限制的信号来取代原先方案中低 TTL 的信号，解决了原先由于 TTL 周期预测不准确带来的种种问题，并避免了 ICMP 发生的可能。而通过引入竞争机制，进一步提高了 NAT 穿越的成功率。

表 3-5 改进前后的比较

	原方案	改进后的方案
TTL 预测	需要	不需要
ICMP 错误	会导致	不会导致
RST 错误	不会导致	会导致

3.3 对称 NAT 的穿越

在试验中发现对称性类型的 NAT 占了 NAT 总数的 10%以上，在某些政府机关，或者大型企业这一比率也许更高。所以想要提高 NAT 穿越的成功率必须就解决对称性 NAT 的穿越问题。

由此可见，映射的端口号是可预测的。

3.3.1 端口预测的方法

终端向服务器分别发送两次连接请求，从两次连接请求的响应消息中可以得到客户端在 NAT 上映射的外网地址，和端口号 Port1 和 Port2。根据这得到的这个两个端口号，计算出端口分配间距 δP= Port2-Port1。预测出终端实际建立连接时将要被分配的端口号 P=Port_Last+ δP(Port_Last 为最后一次映射的端口号)。

终端根据服务器返回的响应消息得到此次报文传送在NAT上映射的IP地址，TCP 连接序号，预测得到的端口等数据。并根据这些数据来进行 NAT 穿越。

那么我们可以写出伪代码如下：

```
if("Symmetric"==GetNATType()){//判断 NAT 类型
    for (int i =0;i<2;i++){//向服务器发送两次请求
    Send Request to Server i;
    Get Response i from Server;//连接信息
    if(response i = Error_Message) //发送失败 返回
```

```
{return -1;}
Get Address i and Port i from response i;//获取地址和端口号
}

if(Address1 !=Address2)//地址不同返回
{return -1;}

if(Port1!=Port2){
δ P=Port2-Port1;//计算端口分配距离
Port=Port2+ δ P;//计算预测端口号
Send Address1， Port and δ P to Server；//交换相关信息
Get Address_Target， Port_Target and δ P_Target to Server；//交换相关信息
for(i=0;i<5;i++){//尝试 5 次

Send packet to Address_Target， Port_Target+ δ P_Target*i; //向预测端口发送信息
Get packet from Address_Target， Port_Target+ δ P_Target*i;
if("Success"==packet.info)
{return 0;)//连接成功 否则重试
}// END for
return -1;//重试后仍未成功，尝试失败
}//END if
```

3.3.2 端口预测可能遇到的问题

基于预测的端口分配策略中存在临界时间段和端口分配范围的问题。临界时间段是指客户端向服务器发送连接请求消息开始到客户端想目的的主机发送连接请求数据包为止所经历的时间段。假设终端 A1 和终端 A2 都是处于 NAT A 后，假如 A1 在时刻 t0 向服务器发送了连接请求消息，并根据服务器返回的响应消息获得客户端在外部的映射地址和可能的端口分配号，随后 A2 在时刻 t1 某个 NAT 外的远程主机发送了一个消息。而当这个消息通过 NAT 时可以能占用了服务器为 A1 预测的端口号，当 A1 在时刻 t2 根据刚才预测到的端口号进行 NAT

穿越时会发生错误，因为这个端口已经被 A2 占用了。

端口分配范围问题则是指的 NAT 在对端口进行分配时，它所分配的端口是在某一个范围之内的比如(0xa000 到 0xafff)。这些端口是循环分配的，当最后一个端口分配完时，会从端口首部继续开始分配。

总结来说 端口预测失败有一下两点常见的原因:

(1)客户端发出连接请求的时段 t0 和 客户端真正发出连接的时段 t2 之间有另一个连接通过了 NAT

(2)客户端发出连接请求时，端口达到了端口范围底部。造成预测失败。

3.3.3 提高端口预测的命中率

首先选通过优化算法等方法减少 δ t=t2-t0 的时间，也就减少了这段时间内出现另一个连接的几率。其次可以使用多端口预测策略，既是在预测端口的时候，不仅仅预测一个端口而是预测一系列端口，当创建一个新的连接时，从这个预测列表中顺序选取一个来尝试，如果无法建立则使用下一个列表中下个端口。知道连接成功或者列表中所有端口都尝试结束。在预测时假如最后一个端口号为 Port_Last,端口间隔为 P,哪么可以建立一个列表 Port_Last+ 1x δ P , Port_Last+ 2x δ P .... Port_Last+ n x δ P(n 一般可以取 4)。通过这种方法可以提高端口预测的命中率。

3.4 本章小节

本章介绍了现有的 NAT 穿越技术包括 NAT/ALG,MIDCOM,STUN,TURN,代理，STUNT,NATBlaster,P2P NAT 等等，并对这些技术的特点，实现原理，以及优缺点进行了分析和比较。最终选用了 STUN(Simple Traversal of UDP Through NAT)协议作为 UDP 部分的穿越方式，而 TCP 部分则是使用基于 NUSS 小组的 STUNT 协议开发。由于基于 STUNT 实现的 TCP 穿越方法依然着几个潜在的问题，本章针对此这些问题进行了更深入的研究，针对 STUNT 模型进行了改进，通过不适用低 TTL 信号和引入竞争提高了 NAT 穿越的成功率。然后我们通过加入端口预测算法，解决了 STUN 和 STUNT 协议中都没有涉及到的对称性 NAT 穿越的问题。

第四章 VNC 穿越 NAT 解决方案

4.1 总体架构

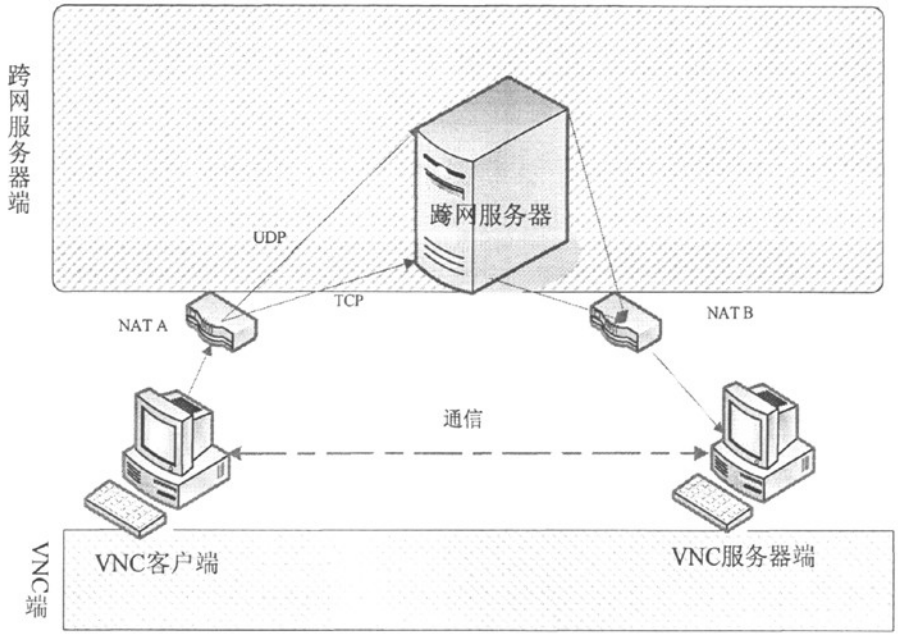


图 4-1 总体架构图

如图 4-1 所示 在公网上部署一台跨网服务器，终端 A 上的 VNC 程序在开始服务时向跨网服务器提交 UDP,或者 TCP 的连接请求。 然后跨网服务器根据请求，尝试进行 NAT 穿透，在穿透成功后。终端 A 和 终端 B 将可以脱离跨网服务器独立进行 UDP 或者 TCP 通信，VNC 程序既可正常工作。而这时跨网服务器仅对连接的维持等状态进行维护。

其中 VNC 客户端和 VNC 服务端属于 VNC 端，而跨网服务器属于跨网服务器端。

4.2 模块汇总

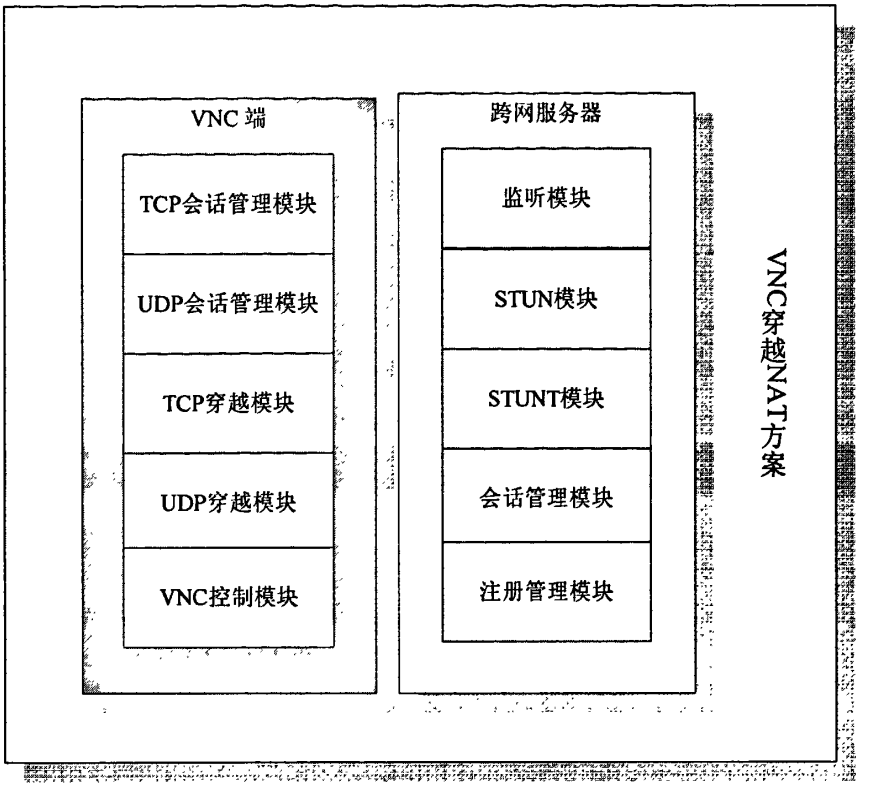


图 4-2 模块汇总图

如图 4-2 所示，这个方案分为两部分 一是对 VNC 端(客户端和服务端)的改造，使其可以进行 NAT 穿越，主要包括 TCP/UDP 会话管理模块，和 TCP/UDP 穿越模块。 第二部分为构建跨网服务器。跨网服务器主要由 UDP 穿越模块（STUN 模块），TCP 穿越模块(STUNT 模块)，和会话管理模块组成。

TCP 会话管理模块负责尝试建立 TCP 连接,在连接失败的时候调用 TCP 穿越模块。UDP 会话管理模块负责尝试建立 UDP 连接，在连接失败的时候调用 UDP 穿越模块。TCP 穿越模块负责进行 TCP 部分的 NAT 穿透，UDP 穿越模块负责 UDP 部分的 NAT 穿透。VNC 控制模块负责控制 VNC 程序的开始与结束，并负责将建立好的连接作为参数传入。

监听模块负责监听 VNC C/S 端发来的各种信息，注册模块负责对 VNC C/S 端进行登记。STUN 模块负责协同 UDP 穿越模块进行 NAT 穿越。STUNT 模块负责协同 TCP 穿越模块进行 NAT 穿越。会话管理模块负责对已经建立的连接进行登记管理。

4.3 工作流程

4.3.1 VNC 端行为

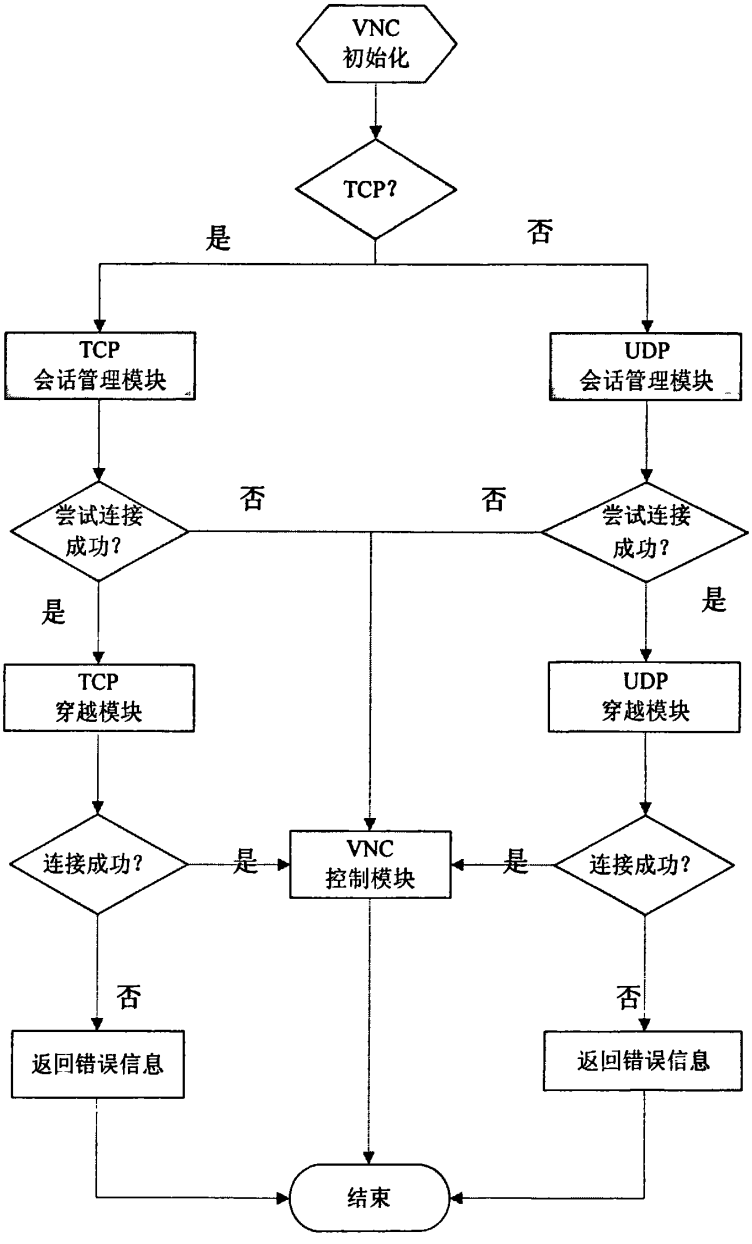


图 4-3 VNC 端工作流程图

如图 4-3 所示 VNC 在初始化后 根据所需连接类型的不同转入相应的会话管

理模块，TCP 转入 TCP 会话模块，UDP 转入 UDP 会话模块，VNC 客户端的会话模块会负责尝试发送连接请求到 VNC 服务端。如果连接成功哪么转入 VNC 控制模块开始正常工作。如果连接失败说明需要进行 NAT 穿透。转入 TCP 或者 UDP 穿透模块。

穿透模块负责和跨网服务中相应的模块协作，尝试穿透 NAT 建立连接。具体的工作流程在后文中有详细描述。当穿透模块完成穿透任务后，如果成功那么就可以转向 VNC 控制模块开始正常工作。失败的话则返回错误信息。本次连接尝试失败。

VNC 控制模块负责讲建立好的 TCP 以及 UDP 连接作为参数传入 VNC 中，并管理 VNC 的开始与结束。

4.3.1.1 UDP 穿越模块

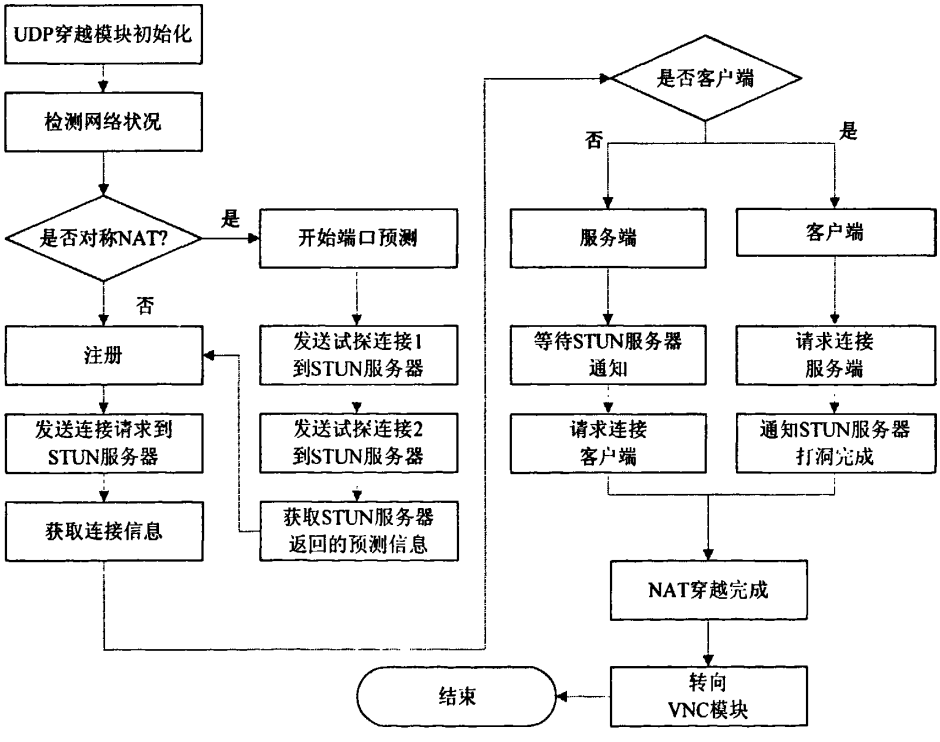


图 4-4 UDP 穿越模块流程图

如图 4-4 所示,UDP 穿越模块初始化后开始进行网络状况检测，如果发现是对称性 NAT 那么进入端口预测过程，通过向 STUN 服务器发送测试连接的方法获

得返回的预测地址与端口然后向 STUN 注册。如果不是对称性 NAT 那么直接发送注册消息到 STUN 服务器。 模块向 STUN 服务器发送连接请求， STUN 服务器会返回所需的连接信息。接着根据模块处于客户端或者服务端进入两条不同的流程，首先客户端向服务端发送连接请求，当连接请求被服务端 NAT 阻挡后，通知 STUN 服务器打洞完成。 STUN 服务器在收到客户端的打洞完成消息后向服务端连接通知，服务端接收到这个消息再反向同客户端进行连接。

当 NAT 穿越完成后转向 VNC 控制模块。

4.3.1.2 TCP 穿越模块

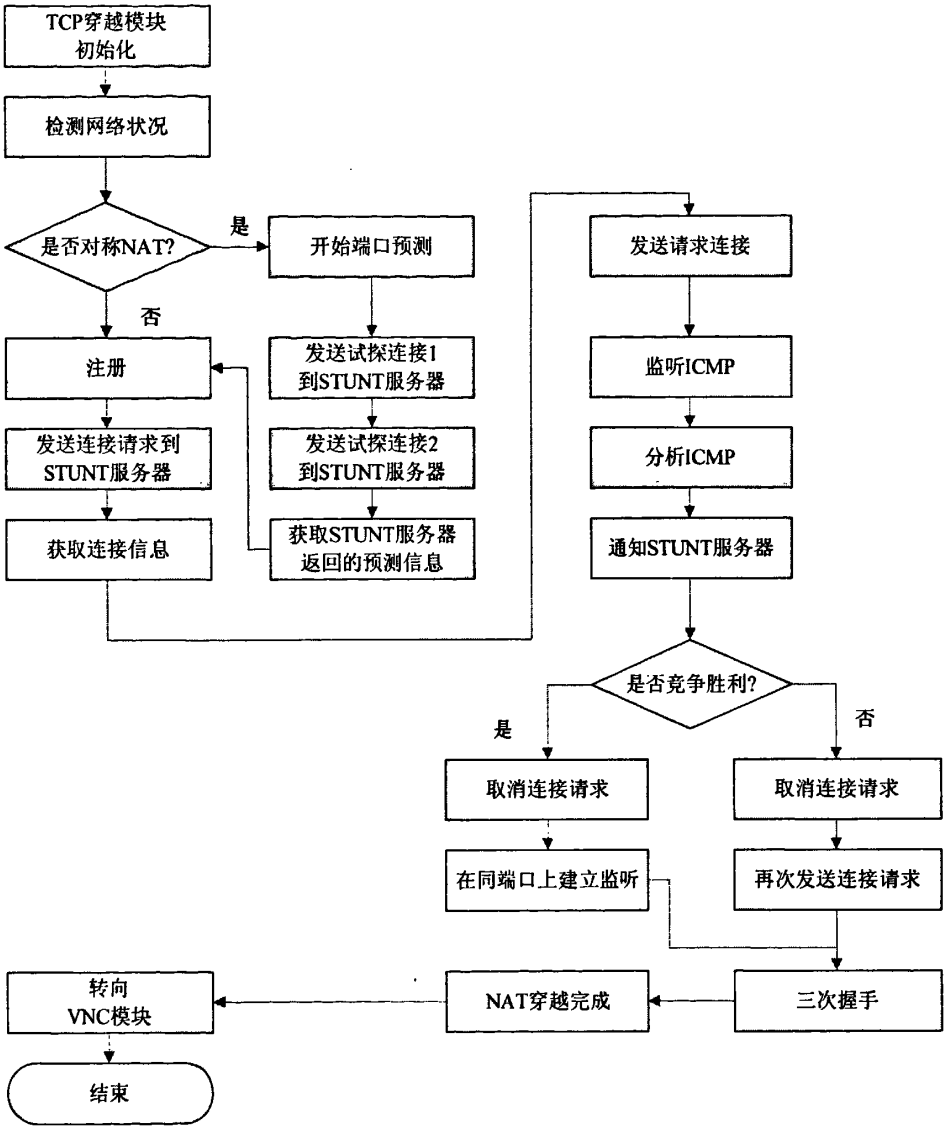


图 4-5 TCP 穿越模块流程图

如图 4-5 所示，TCP 穿越模块初始化后开始进行网络状况检测，如果发现是对称性 NAT 那么进入端口预测过程，通过向 STUNT 服务器发送测试连接的方法获得返回的预测地址与端口然后向 STUNT 注册。如果不是对称性 NAT 那么直接发送注册消息到 STUNT 服务器。注册完毕后模块想 STUN 服务器发送连接请求， STUNT 服务器会返回所需的连接信息。

获得所需连接信息后立刻向对方发送连接请求，并开始监听 ICMP 消息，当 ICMP 消息到达后对其进行分析，获取其中 TCP 连接序号等有用信息并通知 STUNT 服务器。并等待 STUNT 服务器返回竞争结果。 胜利方取消原有的连接尝试，在相同的端口上建立监听，失败方同样取消原有连接，根据 STUNT 服务返回的胜利方相关信息再次进行连接。

在经过正常的 TCP 三次握手 NAT 穿越完成,转向 VNC 控制模块。

4.3.2 跨网服务器端行为

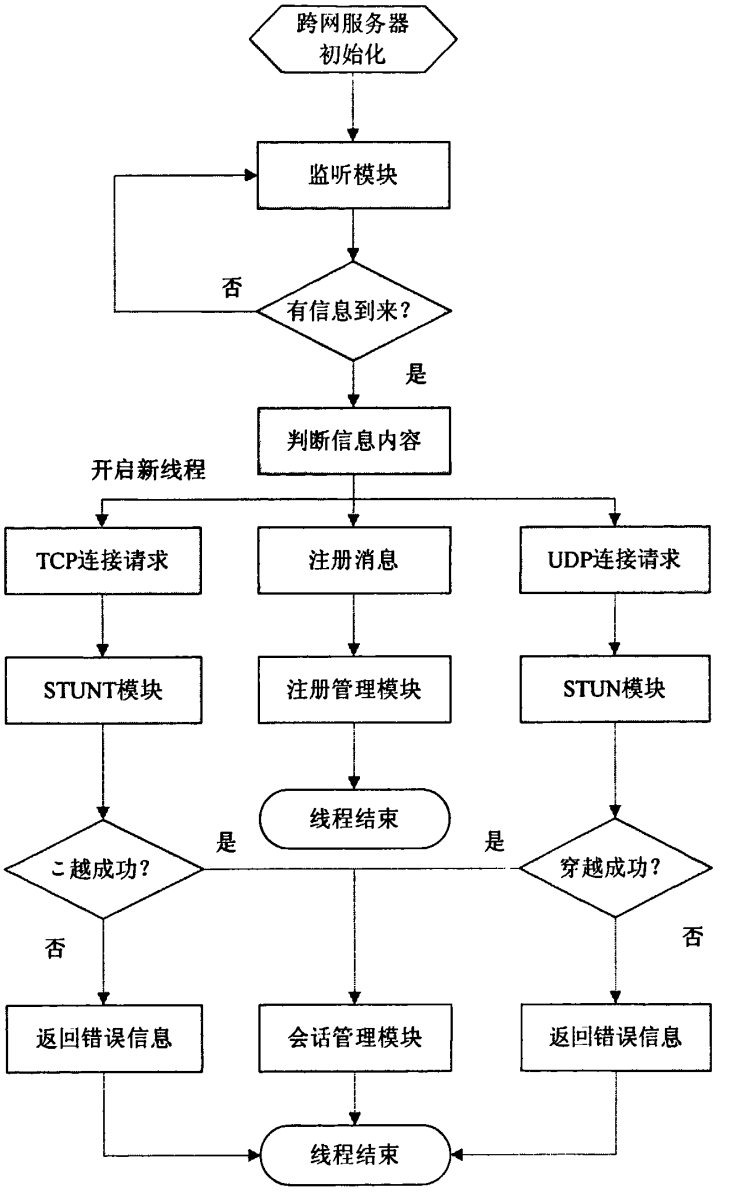


图 4-6 跨网服务器端工作流程图

如图 4-6 所示 当跨网服务器初始化后会激活监听模块，监听模块负责对 VNC C/S 发出的信息进行监听。监听模块在监听到信息后，后会根据所监听到的信息种类的不同，开启一个新的线程并转入相应的模块开始工作。如果收到的是 UDP 连接请求则会转到 STUN 模块，如果接收到的是 TCP 连接请求则会转到

STUNT 模块。如果是注册信息那么转入注册管理模块，在登记完毕后线程结束。如果 STUN 模块或者 STUNT 穿越成功，那么会转入会话管理模块。会话管理模块负责对当前建立的连接进行登记，管理，当会话管理模块处理完毕后会结束这个线程。如果 STUN 模块或者 STUNT 穿越失败哪么则返回错误信息，并结束这个线程。

4.3.2.1 STUN 模块

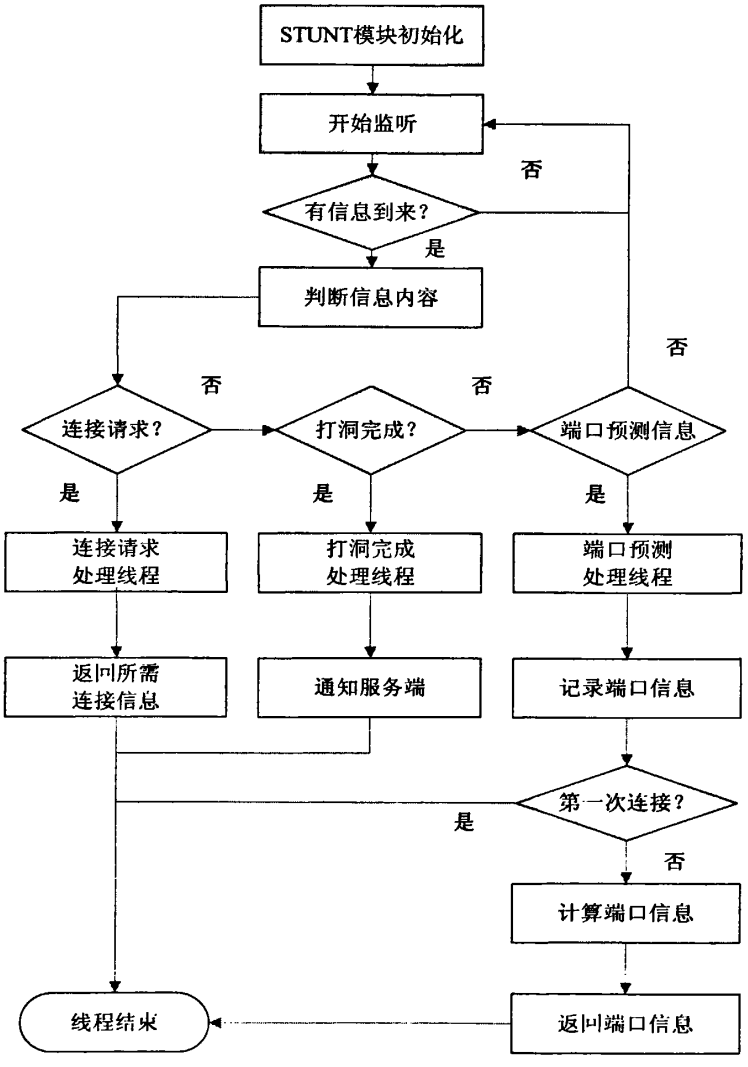


图 4-7 STUN 服务器工作流程图

如图 4-7 所示, STUN 服务器模块初始化后便开始监听网络信息，当信息到来

后根据信息的不同新开启一条线程来处理信息。 如果信息为请求连接，那么查询注册管理模块返回所需的连接信息后结束线程。 如果信息为打洞完成信息，那么通知相应的服务端可以开始连接了，之后结束线程。如果信息为端口预测信息，那么对端口号进行记录，如果是第一条信息那么结束线程，如果已经是第二条以上的信息那么对端口信息进行计算并且返回端口信息，之后结束线程。

4.3.2.2 STUNT 模块

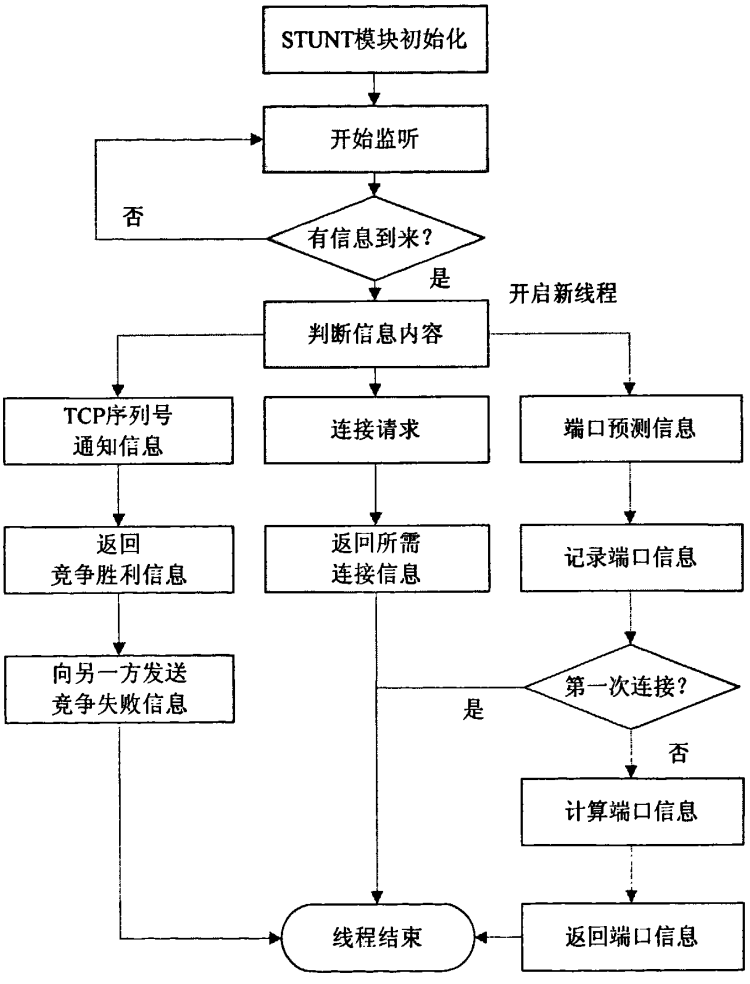


图 4-8 STUNT 服务器工作流程图

如图 4-8 所示，STUNT 服务器模块初始化后便开始监听网络信息，当信息到来后根据信息的不同新开启一条线程来处理信息。 如果信息为请求连接，那么查询注册管理模块返回所需的连接信息后结束线程。 如果信息为 TCP 序列号通

知信息那么通知胜利放弃连接后在相同的端口上新建一个监听，之后通知失败方胜利方所提供的连接信息，通知失败方放弃连接并重新连接胜利方，之后结束线程。如果信息为端口预测信息，那么对端口号进行记录，如果是第一条信息那么结束线程，如果已经是第二条以上的信息那么对端口信息进行计算并且返回端口信息，之后结束线程。

4.4 本章小节

本章提出了一套完整的 VNC 穿越 NAT 解决方案，并详细描述了整体方案的构架及方案中的各个模块的功能。对方案的工作流程进行了也进行了详细的描述。

第五章 实验结果及数据

5.1 实验环境部署

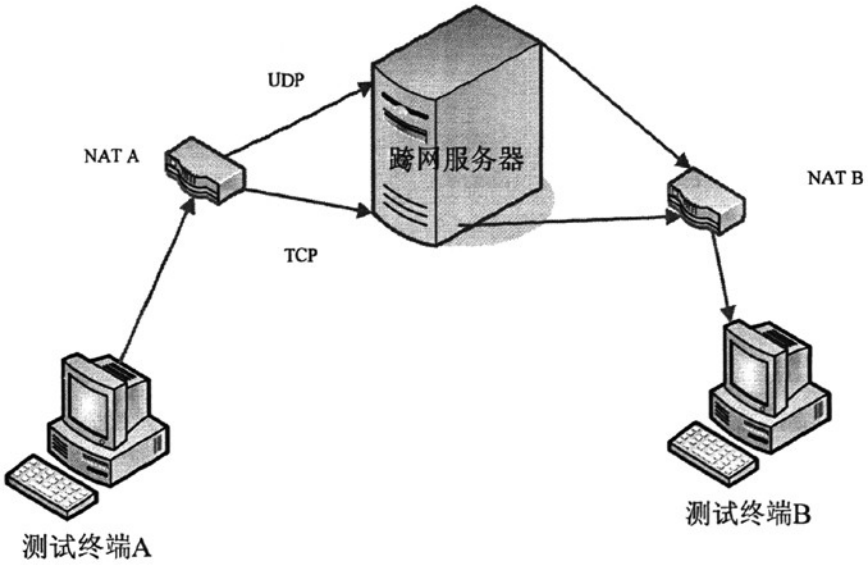


图 5-1 实验环境部署图

如图 5-1 所示，跨网服务器部署在公网上。跨网服务器上装有跨网模块服务器端,包括改进前后的各个版本。

测试终端 A 位于 NAT A 后的私网内，测试终端 B 位于 NAT B 后的私网内内。测试终端 A,B 上装有跨网模块客户端，包括改进前后的各个版本。

我们分别在北航国产化软件实验室，大兴市公安局，以及石景山区市政府部署了测试环境，并进行了测试。 其中用于测试 NAT 数量如表 5-1 所示

表 5-1 测试 NAT 数量表

实验地点	用于测试 NAT 数量
北航国产化软件实验室	28 台
大兴市公安局	13 台
石景山区市政府	17 台

为了对比改进前后 STUNT 的穿越性能对比，在 TCP 部分我们使用了 NUSS 小组提供的 STUNT 协议开源实现包 “XSTUNT”来作为原始数据。同时用于测试

的方案还包括我们自己实现的 STUNT 方案这里叫做原始方案。和以及不使用低 TTL 信号，及引入竞争后两方面改造后的 STUNT 方案，最后还有加入了端口预测的最终方案。

测试时使客户端 A,B 分别通过 NAT 穿越模块向对方开始一次进行一次 UDP 连接和一次 TCP 连接。测试客户端 A,B 各发起 1000 次连接后，统计连接成功的次数。

表 5-2 参与测试的方案列表

参与测试的方案	说明
XSTUNT	XSTUNT 是由 NUSS 小组提供的一个开源 C++函数库，实现了 STUNT 协议中的模型#2，遵守 GPL 协议，
原始方案	由我们自己实现的 STUNT 协议，同样采用了 STUNT 协议中的模型#2。
不使用低 TTL 信号	使用无 TTL 的信号取代低 TTL 的信号。
引入竞争	将竞争引入 STUNT 模型，通过双方各尝试一次的方法来提高成功率。
使用端口预测	使用端口预测的方法穿越对称性 NAT。

5.2 实验结果及数据

我们把终端 A 的 IP 地址设置为 192.168.2.100 和终端 B 的 IP 地址 192.168.3.100 分别属于两个 NAT 后，其中 192.168.2.100 所在的 NAT A 的公网 IP 为 192.168.1.102，而 192.168.3.100 所在的 NAT B 的公网 IP 为 192.168.1.101

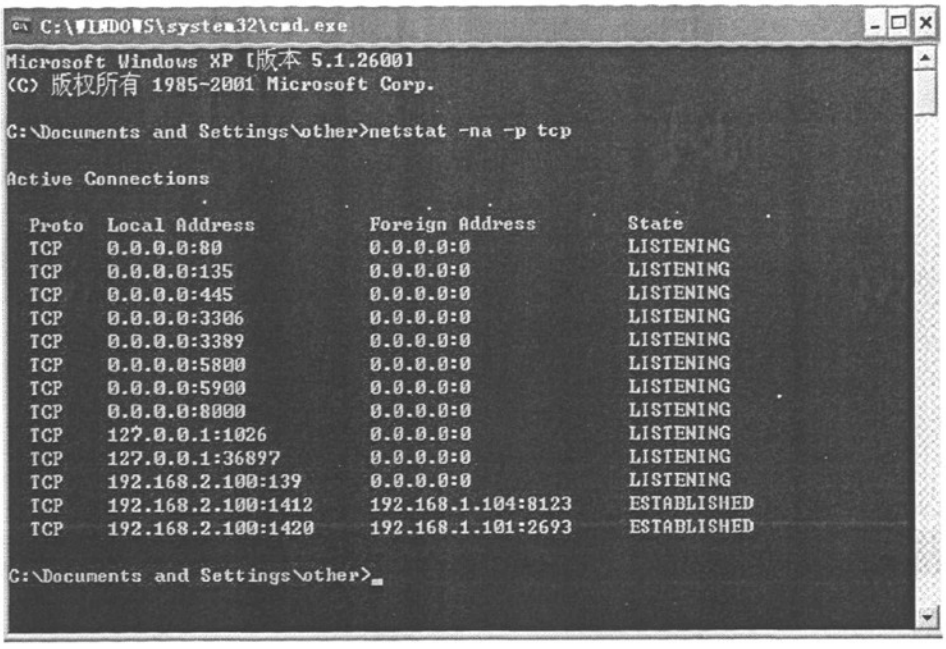


图 5-2 终端 A 连接状态图

经过跨网服务器后终端 A 的连接情况，通过图 5-2 我们可以看到，它同 192.168.3.100 所在的 NAT 的公网 IP：192.168.1.101 建立了连接

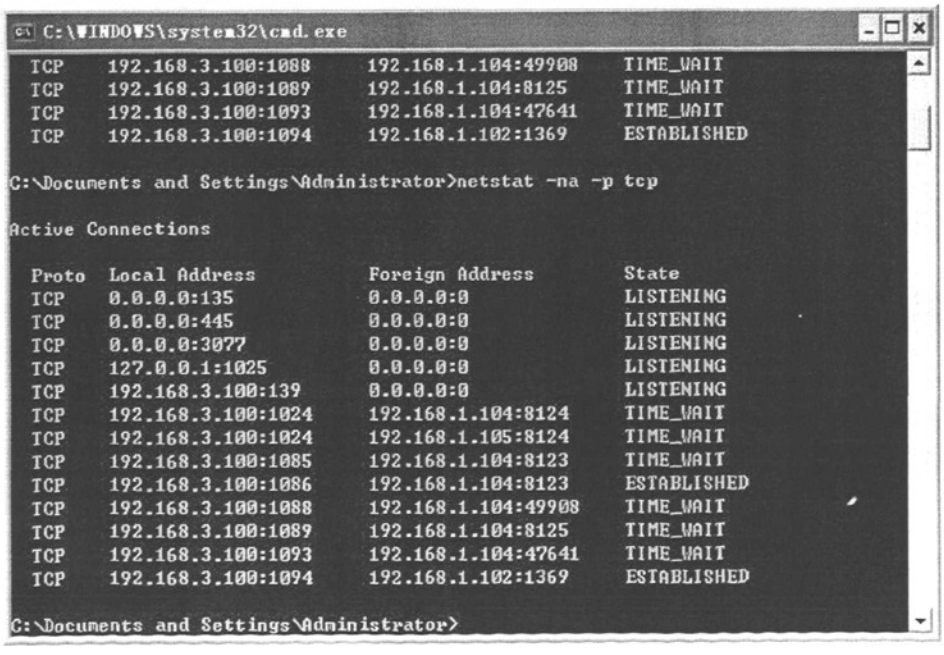


图 5-3 终端 B 连接状态图

终端 B 的连接情况，通过图 5-3 我们可以看到它同 192.168.2.100 所在的 NAT 的公网 IP：192.168.1.102 建立了连接。结合上面两点我们可以认为终端 A 和终端 B 成功建立了连接。实验的测试结果如下表 5-3 所示。

表 5-3 测试结果表

使用方案	尝试次数	UDP 成功次数	TCP 成功次数	总成功比率
XSTUNT	2000	1782（89%）	1489 (74%)	81.5%
原始方案	2000	1783（89%）	1522 (76%)	82.5%
不使用低 TTL	2000	1784（89%）	1586（79%）	84%
引入竞争	2000	1782（89%）	1643 (82%)	85.5%
使用端口预测(最终方案)	2000	1996（99%）	1825 (91%)	95%

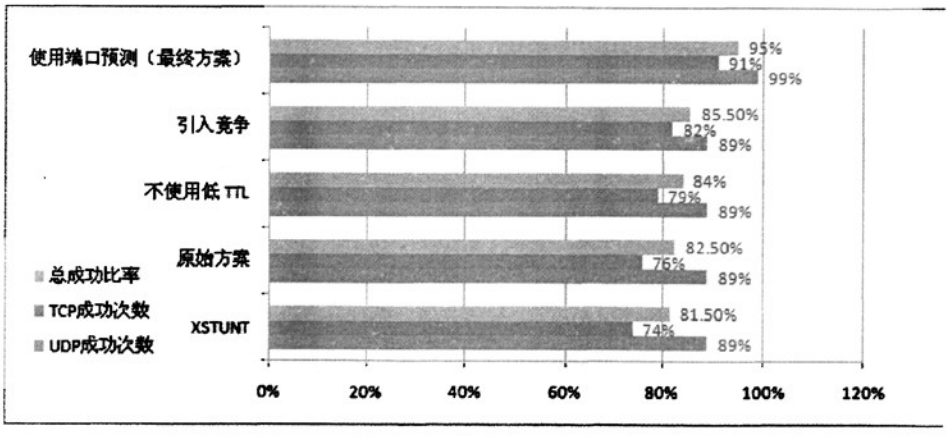


图 5-4 测试结果柱状图

从表 5-3 和图 5-4 中我们可以看到原始方案即我们自己实现的 STUNT 方案和和 NUSS 提供的 XSTUNT 的成功率基本相同，相差仅为 1%，差距不大。经过改进后的 STUNT 方案在引入竞争和不使用低 TTL 后都让 TCP 部分的穿越成功率有 3%-6%的提升。由于改进方案的前两部并未设计 UDP 部分所以总的成功率只提成了 2%-3%。

最终引入端口预测后 UDP 部分和 TCP 部分的成功率都有大幅度的提升，UDP 部分提升了 10%而 TCP 部分则提升了 9%。这也正好符合第四章中对称性 NAT

在 NAT 总数中占有 10%左右的比率的论述。

从上面的测试结果我们可以看到经过改进后的最终方案的综合成功率达到了 95%比起 81.5% (XSTUNT)和 82.5%(我们自己实现的原始方案)有了非常大的提升，而其中 UDP 穿越更是达到了 99%的成功率。已经完全可以达到应用的要求。

第六章 总结与展望

本文主要针对现有 NAT 穿越方案进行了研究和改进，并在此基础上提出了一个综合的 VNC 穿越 NAT 整体方案，NAT 技术是为了解决 Internet 上 IPv4 地址日益紧张的状况而提出的一种网络技术。NAT 网关设备通过修改内部网络数据包的包头 IP 地址信息并跟踪该映射信息，使同一局域网内的多个终端节点可以共享同一个外部 IP 地址，从而在一定程度上缓解了 IPv4 地址紧张的问题。NAT 在 Internet 上的大规模使用也为现有的网络应用，尤其是对等应用带来了很多问题。由于 NAT 技术改变了网络数据包的地址信息，使得对等连接无法成功建立，因此许多对等应用无法运行在 NAT 网关设备存在的环境中。为了解决这个问题，业界提出了不少相应的解决方案如 STUN，TURN，NATBlaster,STUNT 等，但这些解决方案在使用过程有各自的优点和不足。

本文首先讨论了不同 NAT 穿越解决方案的差别，接着选择了其中的 STUN 协议和 STUNT 协议分别作为 UDP 部分和 TCP 部分的解决方案。STUN 和 STUNT 的共同问题在于无法处理对称性 NAT 的穿越。因此在我们的改进方案中我们使用了端口预测技术，端口预测技术利用对称 NAT 端口分配的规律性进行预测从而实现对对称性 NAT 的穿越。同时由于 STUNT 在进行 TCP 穿越时也存在着不少的问题。如预测 TTL 不准确引发各种错误，ICMP 早于 TTL 前返回被 NAT 当作致命错误拦截等问题。针对这些问题我们提出了不适用低 TTL 信号，引入竞争等方法来进一步提高穿越成功率。这些改进通过第四章的实验数据得到了很好的验证。根据实验结果及数据，最终的的 NAT 穿越方案达到了 95%的总穿越成功率，已经完全能够满足一般应用的需求。

但是本文的研究在某些方面还是有限的。第一，我们没有测试所有现存的 NAT 设备。我们研究只涉及了相对较少的 NAT 设备，仅仅局限于最近北京市场上常见的 NAT 种类,对于以前的老型 NAT 没有进行测试，而在本文撰写的过程中也不断有新型号的 NAT 投入市场。第二，由于条件限制，我们测试的范围比较小，仅限于北航实验室，大兴公安局，以及石景山区政府等地。第三，我们只测试了家用 NAT。大型企业网络中所运用的 NAT 的端口预测成功率通常会更低一些，第四，尽管 TCP 穿透 NAT 技术也可以广泛地用于穿透防火墙，但我们并没有实际测试 NAT 坏境之外的防火墙。另外我们也没有测试基于 IPv6 网络。

但是我们相信随着对 NAT 穿越技术的深入了解和认识，新的，更好的 NAT 穿越方案会不断涌现。另一方面随着 IPv6 的日益普及，IP 地址短缺的问题将有望被彻底解决，NAT 网关也很可能会慢慢退出历史的舞台。

参考文献

[1] Richardson T,Stafford-Fraser Q,Wood KR,et al. Vritual Network Computing[J].IEEE Internet Computing,1998,2.

[2] 孙盛源.简述 NAT-网络地址转换 甘肃科技 2001，(5).

[3] Srisuresh P, Tsirtsis G, Akkiraju P, et al. DNS extensions to Network Address Translators (DNS_ALG) .RFC 2649. 1999 .

[4] Srisuresh P, Kuthan J, Rosenberg J, Molitor A, Rayhan A, Middlebox communication architecture and framework, RFC 3303, Internet Engineering Task Force,August 2002

[5] Rosenberg, J., Weinberger, J., Huitema, C., and Mahy, R. RFC 3489: STUN — Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), Mar. 2003.

[6] Rosenberg, J., Mahy, R., and Huitema, C. Internet draft: TURN — Traversal Using Relay NAT, Feb. 2004.

[7] 张巍. 基于 Full Proxy 的 H.323 协议穿透 NAT 解决方案[J]. 电脑知识与技术(学术交流), 2007,(16) .

[8] Hain,T.,Architectural Implications of NAT,RFC 2993,IETF,November 2000.

[9] Rosenberg, J., Weinberger, J., Huitema, C., and Mahy, R. RFC 3489: STUN — Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), IEEE Compute Soicety Press, Mar. 2003.

[10] Srisuresh P,et,al.IP Network Address Translator(NAT) Terminology and Considerations[S] RFC 2663,IETF issue Aug.1999.

[11] Guha, S., Takeda, Y., and Francis, P. NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity. In Proceedings of SIGCOMM'04 Workshops (Portland, OR, Aug. 2004), pp. 43—48.

[12] Saikat and Paul Francis. Simple traveral of UDP Through NATs and TCP too, IEEE Compute Soicety Press,Deceber 11,2004

[13] Biggadike, A., Ferullo, D., Wilson, G., and Perrig, A. NATBLASTER: Establishing TCP connections between hosts behind NATs. In Proceedings of ACM SIGCOMM ASIA Workshop (Beijing, China, Apr. 2005).

[14] Doyle J.TCP/IP 路由技术(第 2 卷)[M].北京:人民邮电出版社,2003.

[15] Eppinger, J. L. TCP Connections for P2P Apps: A Software Approach to Solving the NAT Problem. Tech. Rep. CMU-ISRI-05-104, Carnegie Mellon University, Pittsburgh, PA, Jan. 2005.

[16] Guha, S. STUNT - Simple Traversal of UDP Through NATs and TCP too. [online] http://nutss.gforge.cis.cornell.edu/pub/draft-guha-STUNT-00.txt

[17] Guha S,Francis P. Characterization and Measurement of TCP Traversal Through NATs and Firewalls[C] .Proc.of Internet Measurement Conference. :2005-10 .

[18] Takeda, Y. Internet draft: Symmetric NAT Traversal using STUN, June 2003.

[19] Nordmark E. Stateless IP/ICMP Translation Algorithm (SIIT) .RFC 2765. 2000,

[20] 高扬,肖继民. NAT 穿越技术研究[J].江苏通信技术, 2005,(05) .

[21] 刘小军,周文科.跨越 NAT 的 P2P 应用 UDP 通信[J].现代计算机,2006,(01).

[22] 谭峻松,庞采哲,首照宇.NAT 网络穿越技术的研究[J].微计算机信息,2006,(9).

[23] 何宝宏.浅析 NAT 的类型[J].电信网技术,2004,(8).

[24] 刘敏,曾明,陈建明,原盛.NAT 技术的分类及识别策略[J].计算机工程与应用,2002,(10).

研究生期间撰写论文

[1] 庄霄，邓中亮 “基于 STUNT 的 TCP 穿越 NAT 技术研究”，中国科技论文在线，2 月 9 日。

致谢

研究生阶段的学习生活就要结束了，在这两年半时间里，我所收获的不仅仅是学到的专业知识，更多的是从各位老师和同学身上看到的求是创新精神。

首先，我要感谢我的导师---邓中亮老师。在研究生阶段，他们一直在各方面给予我指导和关怀，使我能够顺利地完成毕业设计，他们严谨的治学态度、渊博的学识、敏锐的洞察力以及对学生热情的指导鼓励，都给我留下了难忘的记忆，必将终生受益！

同时，还要感谢与我共同渡过这一段难忘时光的刘旭文，梁京等同学，他们勤奋刻苦、认真求实的工作态度，深深激励着我；感谢我的父亲和母亲，他们给予我的爱和支持是我不断前进的动力！

最后，感谢各位专家、评委和老师百忙之中抽出宝贵的时间评阅本论文！