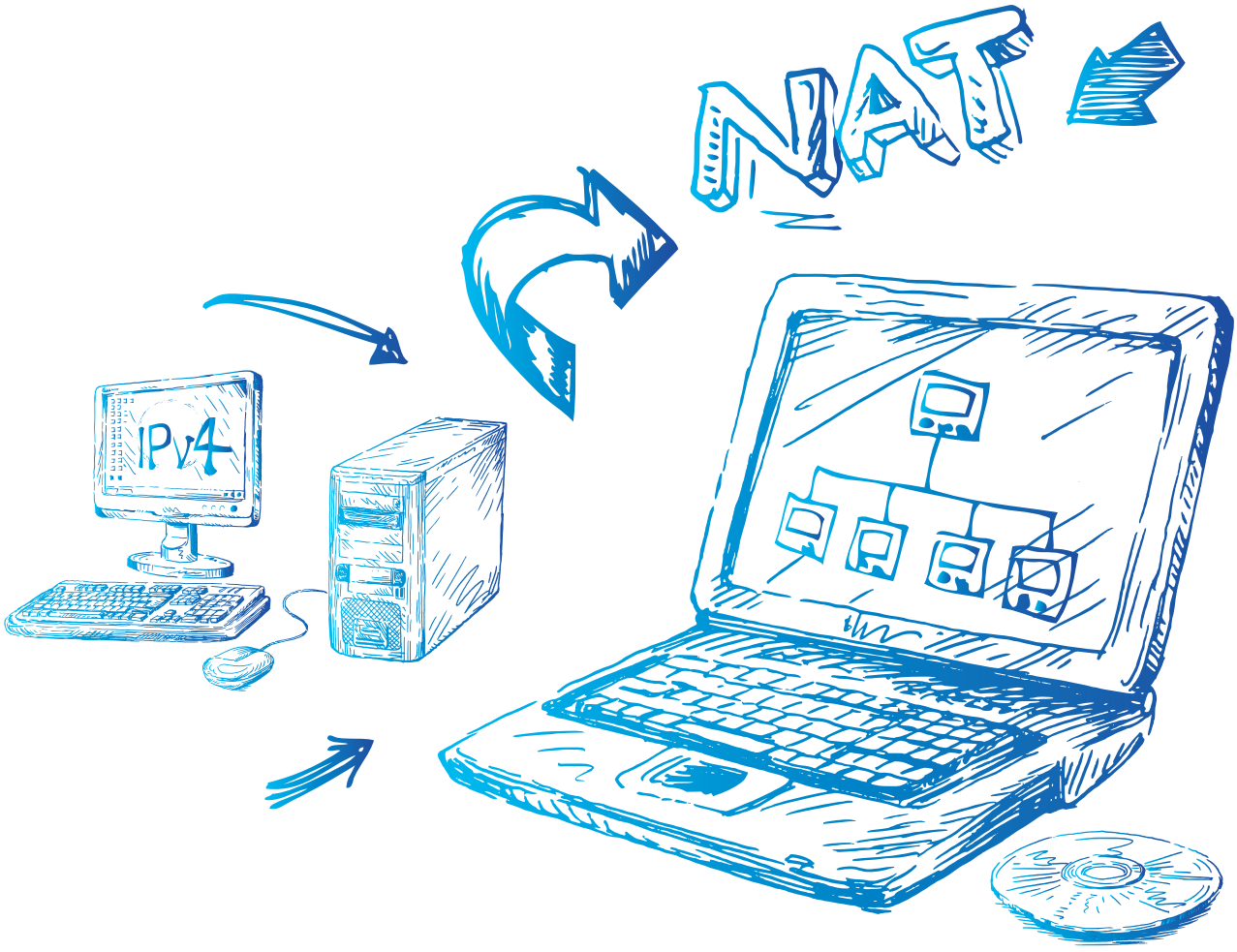
NAT专题

网络大爬虫
Network Addicts

2012年第1期总第5期

H³Care俱乐部VIP客户专属



NAT概述 P001
NAT转发一些细节的汇总 P010
NAT的特殊处理 P018
NAT的双机热备方案 P078
NAT典型组网实例 P083
NAT-PT介绍 P091

H3C

编者寄语

小L的NAT机器

在浩瀚无边的太平洋中部，有一个遗世而独立的弹丸小国，美其名曰桃花岛，这是一个由IP公民构成的和谐小国，岛上登记在册的公民共有254位，小L便是芸芸众生中的一员。

对于IP公民而言，每个人都需要1个IP地址，否则寸步难行。IP地址分为私有和公有两类，桃花岛虽然人口过百，但是国际地位不济，联合国只授予1个公有IP地址，这还是低声下气讨来的。

面对公有IP极度贫乏的现状，桃花岛实行私有IP制度，每个公民分得1个私有IP地址，可以在岛内畅行无阻，但是不能踏出国门半步，否则下场就是有去无回，整个岛国只有总统配备公有IP地址，既可以在岛内活动，也可以出国访问。

由于极少与外界往来，桃花岛丰富的海产资源不能出口，岛内经济一直停滞不前，为了提升国民幸福感，总统于是布告天下，谁能在30日之内解决私有IP地址出境访问难题，奖励500万大洋，并赐封工信部长要职，重赏之下必有勇夫，岛内有识之士皆跃跃欲试。

小L家境困难，常常食不果腹，听说有此好事，暗下决心，志在必得。自此小L起早贪黑，反复研读Routing TCP IP宝典，省吃俭用购买实验器材，功夫不负有心人，不出一个月，小L终于制作出样机一台，并将其命名为NAT机器，地址转换之意。

小L踌躇满志的抱着样机奔赴总统府邸，不过总统并没有对小L寄予厚望，几天来总统已接见多位小L之流的民间高手，这些所谓的高手最终都铩羽而归。小L在总统面前启动了NAT机器，NAT机器外形如同一个隧道，分为入口和出口，只见小L匍匐着进入NAT机器入口，瞬间又从出口出来，总统惊奇的发现小L的私有IP地址竟然自动变化为一个公有IP，小L紧接着又从出口进去，霎那间回到入口，小L的IP地址又恢复为私有IP，总统彻底为小L神奇的NAT机器所折服，立马命财政部长开具500万支票与小L，并任命小L为工信部长。

小L的NAT机器被部署在桃花岛海关总署，岛内公民通过NAT机器自由出入境，桃花岛的海产品源源不断的输送到世界各地，国外琳琅满目的商品也大批进入桃花岛，岛上公民从此过上幸福的生活，而小L担任工信部长后，开始研究IPv6技术，他的奋斗目标是给每个岛上公民分配1个公有IPv6地址。

许青邦

CONTENTS 目录

2012年第1期总第5期　NAT专题

综　述

NAT概述 001

NAT基础机理

NAT转发一些细节的汇总 010
NAT的特殊处理 018

NAT组网的解决方案

UPnP基本原理以及在NAT中的应用 024
STUN/TURN技术浅析 034
Punching的实现与应用 050
P2P中的NAT穿越方案简介 058
SR8800路由器NAT处理流程 065

涉及NAT组网的相关协议的兼容性处理

NAT　ALG原理与应用 067
传统VPN与NAT穿越的兼容性 072

NAT本身组网

NAT的双机热备方案 078
NAT典型组网实例 083

地址转换的其他方案

NAT-PT介绍 091
H3C防火墙NAT“无限次”转换说明 100

NAT概述

文/杜祥宇

IPv4协议和NAT的由来

今天，无数快乐的互联网用户在尽情享受Internet带来的乐趣。他们浏览新闻、搜索资料、下载软件、广交新朋、分享信息，甚至于足不出户获取一切日用所需。企业利用互联网发布信息、传递资料和订单、提供技术支持、完成日常办公。然而，Internet在给亿万用户带来便利的同时，自身却面临一个致命的问题：构建这个无所不能的Internet的基础IPv4协议已经不能再提供新的网络地址了。

2011年2月3日中国农历新年，IANA对外宣布：IPv4地址空间最后5个地址块已经被分配给下属的5个地区委员会。2011年4月15日，亚太区委员会APNIC对外宣布，除了个别保留地址外，本区域所有的IPv4地址基本耗尽。一时之间，IPv4地址作为一种濒危资源身价陡增，各大网络公司出巨资收购剩余的空闲地址。其实，IPv4地址不足问题已不是新问题，早在20年以前，IPv4地址即将耗尽的问题就已经摆在Internet先驱们面前。这不禁让我们想去了解，是什么技术使这一危机延缓了近20年。

要找到问题的答案，让我们先来简略回顾一下IPv4协议。

IPv4即网际网协议第4版——Internet Protocol Version 4的缩写。IPv4定义一个跨越异种网络互连的超级网，它为每个网际网的节点分配全球唯一IP地址。如果我们把Internet比作一个邮政系统，那么IP地址的作用就等同于包含城市、街区、门牌编号在内的完整

地址。IPv4使用32bits整数表达一个地址，地址最大范围就是2^{32}约为43亿。以IP创始时期可被联网的设备来看，这样的一个空间已经很大，很难被短时间用完。然而，事实远远超出人们的设想，计算机网络在此后的几十年里迅速壮大，网络终端数量呈爆炸性增长。

更为糟糕的是，为了路由和管理方便，43亿的地址空间被按照不同前缀长度划分为A、B、C、D类地址网络和保留地址。其中，A类网络地址127段，每段包括主机地址约1678万个。B类网络地址16384段，每段包括65536个主机地址。

A类地址，网络8位	0*******	--------	--------	--------
B类地址，网络16位	10******	********	--------	--------
C类地址，网络24位	110*****	********	********	--------
D类地址，保留	1110----	--------	--------	--------
以二进制表达，*表示网络位，-表示主机位				

图1 IPv4网络地址划分

IANA向超大型企业/组织分配A类网络地址，一次一段。向中型企业或教育机构分配B类网络地址，一次一段。这样一种分配策略使得IP地址浪费很严重，很多被分配出去的地址没有真实被利用，地址消耗很快。以至于二十世纪90年代初，网络专家们意识到，这样大手大脚下去，IPv4地址很快就要耗光了。于是，人们开始考虑IPv4的替代方案，同时采取一系列的措施来减缓IPv4地址的消耗。正是在这样一个背景之下，本期的主角闪亮登场，它就是网络地址转换——NAT。

NAT是一项神奇的技术，说它神奇在于它的出现几乎使IPv4起死回生。在IPv4已经被认为行将结束历史使命之后近20年时间里，人们几乎忘了IPv4的地址空间即将耗尽这样一个事实——在新技术日新月异的时代，20年可算一段漫长的历史。更不用说，在NAT产生以后，网络终端的数量呈加速上升趋势，对IP地址的需求剧烈增加。因此足见NAT技术之成功，影响之深远。

说它神奇，更因为NAT给IP网络模型带来了深远影响，其身影遍布网络每个角落。根据一份最近的研究报告，70%的P2P用户位于NAT网关以内。因为P2P主要运行在终端用户的个人电脑之上，这个数字意味着大多数PC通过NAT网关连接到Internet。如果加上2G和3G方式联网的智能手机等移动终端，在NAT网关之后的用户远远超过这个比例。

然而当我们求本溯源时却发现一个很奇怪的事实：NAT这一意义重大的技术，竟然没有公认的发明者。NAT第一个版本的RFC作者，只是整理归纳了已被广泛采用的技术。

NAT的工作模型和特点

NAT的概念模型

NAT名字很准确，网络地址转换，就是替换IP报文头部的地址信息。NAT通常部署在一个组织的网络出口位置，通过将内部网络IP地址替换为出口的IP地址提供公网可达性和上层协议的连接能力。那么，什么是内部网络IP地址？

RFC1918规定了三个保留地址段落：10.0.0.0-10.255.255.255；172.16.0.0-172.31.255.255；192.168.0.0-192.168.255.255。这三个范围分别处于A，B，C类的地址段，不向特定的用户分配，被IANA作为私有地址保留。这些地址可以在任何组织或企业内部使用，和其他Internet地址的区别就是，仅能在内部使用，不能作为全球路由地址。这就是说，出了组织的管理范围这些地址就不再有意义，无论是作为源地址，还是目的地址。对于一个封闭的组织，如果其网络不连接到Internet，就可以使用这些地址而不用向IANA提出申请，而在内部的路由管理和报文传递方式与其他网络没有差异。

对于有Internet访问需求而内部又使用私有地址的网络，就要在组织的出口位置部署NAT网关，在报文离开私网进入Internet时，将源IP替换为公网地址，通常是出口设备的接口地址。一个对外的访问请求在到达目标以后，表现为由本组织出口设备发起，因此被请求的服务端可将响应由Internet发回出口网关。出口网关再将目的地址替换为私网的源主机地址，发回内部。这样一次由私网主机向公网服务端的请求和响应就在通信两端均无感知的情况下完成了。依据这种模型，数量庞大的内网主机就不再需要公有IP地址了。

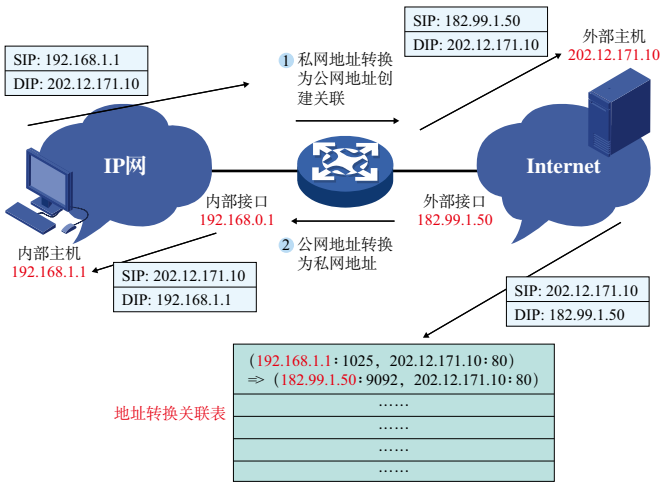


图2 NAT转换过程示意图

虽然实际过程远比这个复杂，但上面的描述概括了NAT处理报文的几个关键特点：

■ 网络被分为私网和公网两个部分，NAT网关设置在私网到公网的路由出口位置，双向流量必须都要经过NAT网关；

■ 网络访问只能先由私网侧发起，公网无法主动访问私网主机；

■ NAT网关在两个访问方向上完成两次地址的转换或翻译，出方向做源信息替换，入方向做目的信息替换；

■ NAT网关的存在对通信双方是保持透明的；

■ NAT网关为了实现双向翻译的功能，需要维护一张关联表，把会话的信息保存下来。

随着后面对NAT的深入描述，读者会发现，这些特点是鲜明的，但又不是绝对的。其中第二个特点打破了IP协议架构中所有节点在通讯中的对等地位，这是NAT最大的弊端，为对等通讯带来了诸多问题，当然相应的克服手段也应运而生。事实上，第四点是NAT致力于达到的目标，但在很多情况下，NAT并没有做到，因为除了IP首部，上层通信协议经常在内部携带IP地址信息。这些我们稍后解释。

一对一的NAT

如果一个内部主机唯一占用一个公网IP，这种方式被称为一对一模型。此种方式下，转换上层协议就是不必要的，因为一个公网IP就能唯一对应一个内部主机。显然，这种方式对节约公网IP没有太大意义，主要是为了实现一些特殊的组网需求。比如用户希望隐藏内部主机的真实IP，或者实现两个IP地址重叠网络的通信。

一对多的NAT

NAT最典型的应用场景就如同图2描述的，一个组织网络，在出口位置部署NAT网关，所有对公网的访问表现为一台主机。这就是所谓的一对多模型。这种方式下，出口设备只占用一个由Internet服务提供商分配的公网IP地址。面对私网内部数量庞大的主机，如果NAT只进行IP地址的简单替换，就会产生一个问题：当有多个内部主机去访问同一个服务器时，从返回的信息不足以区分响应应该转发到哪个内部主机。此时，需要NAT设备根据传输层信息或其他上层协议去区分不同的会话，并且可能要对上层协议的标识进行转换，比如TCP或UDP端口号。这样NAT网关就可以将不同的内部连接访问映射到同一公网IP的不同传输层端口，通过这种方式实现公网IP的复用和解复用。这种方式也被称为端口转换PAT、NAPT或IP伪装，但更多时候直接被称为NAT，因为它是最典型的一种应用模式。

按照NAT端口映射方式分类

在一对多模型中，按照端口转换的工作方式不同，又可以进行更进一步的划分。为描述方便，以下将IP和端口标记为（nAddr:nPort），其中n代表主机或NAT网关的不同角色。

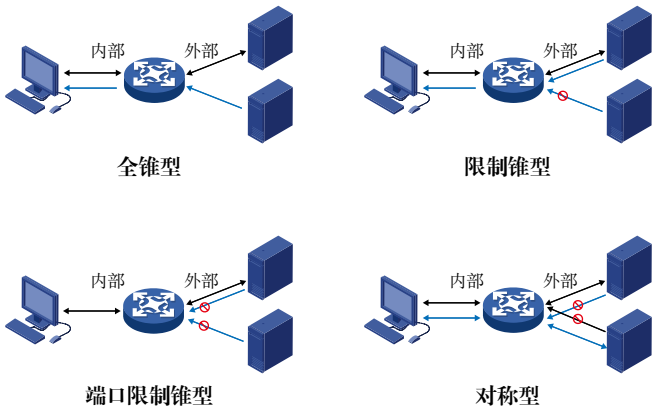


图3 按照端口转换映射方式分类

全锥形NAT

其特点为：一旦内部主机端口对（iAddr:iPort）被NAT网关映射到（eAddr:ePort），所有后续的（iAddr:iPort）报文都会被转换为（eAddr:ePort）；任何一个外部主机发送到（eAddr:ePort）的报文将会被转换后发到（iAddr:iPort）。

限制锥形NAT

其特点为：一旦内部主机端口对（iAddr:iPort）被映射到（eAddr:ePort），所有后续的（iAddr:iPort）报文都会被转换为（eAddr:ePort）；只有（iAddr:iPort）向特定的外部主机hAddr发送过数据，主机hAddr从任意端口发送到（eAddr:ePort）的报文将会被转发到（iAddr:iPort）。

端口限制锥形NAT

其特点为：一旦内部主机端口对（iAddr:iPort）被映射到（eAddr:ePort），所有后续的（iAddr:iPort）报文都会被转换为（eAddr:ePort）；只有（iAddr:iPort）向特定的外部主机端口对（hAddr:hPort）发送过数据，由（hAddr:hPort）发送到（eAddr:ePort）的报文将会被转发到（iAddr:iPort）。

对称型NAT

其特点为：NAT网关会把内部主机“地址端口对”和外部主机“地址端口对”完全相同的报文看作一个连接，在网关上创建一个公网“地址端口对”映射进行转换，只有收到报文的外部主机从对应的端口对发送回应的报文，才能被转换。即使内部主机使用之前用过的地址端口对去连接不同外部主机（或端口）时，NAT网关也会建立新的映射关系。

事实上，这些术语的引入是很多混淆的起源。现实中的很多NAT设备是将这些转换方式混合在一起工作的，而不单单使用一种，所以这些术语只适合描述一种工作方式，而不是一个设备。比如，很多NAT设备对内部发出的连接使用对称型NAT方式，而同时支持静态的端口映射，后者可以被看作是全锥型NAT方式。而有些情况下，NAT设备的一个公网地址和端口可以同时映射到内部几个服务器上以实现负载分担，比如一个对外提供Web服务器的站点可能是有成百上千个服务器在提供HTTP服务，但是对外却表现为一个或少数几个IP地址。

NAT的限制与解决方案

IP端到端服务模型

IP协议的一个重要贡献是把世界变得平等。在理论上，具有IP地址的每个站点在协议层面有相当的获取服务和提供服务的能力，不同的IP地址之间没有差异。人们熟知的服务器和客户机实际是在应用协议层上的角色区分，而在网络层和传输层没有差异。一个具有IP地址的主机既可以是客户机，也可以是服务器，大部分情况下，既是客户机，也是服务器。端到端对等看起来是很平常的事情，而意义并不寻常。但在以往的技术中，很多协议体系下的网络限定了终端的能力。正是IP的这个开放性，使得TCP/IP协议族可以提供丰富的功能，为应用实现提供了广阔平台。因为所有的IP主机都可以服务器的形式出现，所以通讯设计可以更加灵活。使用UNIx/Linux的系统充分利用了这个特性，使得任何一个主机都可以建立自己的HTTP、SMTP、POP3、DNS、DHCP等服务。与此同时，很多应用也是把客户端和服务器的角色组合起来完成功能。例如在VoIP应用中，用户端向注册服务器登录自己的IP地址和端口信息过程中，主机是客户端；而在呼叫到达时，呼叫处理服务器向用户端发送呼叫请求时，用户端实际工作在服务器模式下。在语音媒体流信道建立过程后，通讯双向发送语音数据，发送端是客户模式，接收端是服务器模式。而在P2P的应用中，一个用户的主机既为下载的客户，同时也向其他客户提供数据，是一种C/S混合的模型。上层应用之所以能这样设计，是因为IP协议栈定义了这样的能力。试想一下，如果IP提供的能力不对等，那么每个通信会话都只能是单方向发起的，这会极大限制通信的能力。细心的读者会发现，前面介绍NAT的一个特性正是这样一种限制。没错，NAT最大的弊端正在于此——破坏了IP端到端通信的能力。

NAT的弊端

NAT在解决IPv4地址短缺问题上，并非没有副作用，其实存在很多问题。

首先，NAT使IP会话的保持时效变短。因为一个会话建立后会在NAT设备上建立一个关联表，在会话静默的这段时间，NAT网关会进行老化操作。这是任何一个NAT网关必须做的事情，因为IP和端口资源有限，通信的需求无限，所以必须在会话结束后回收资源。通常TCP会话通过协商的方式主动关闭连接，NAT网关可以跟踪这些报文，但总是存在例外的情况，要依赖自己的定时器去回收资源。而基于UDP的通信协议很难确定何时通信结束，所以NAT网关主要依赖超时机制回收外部端口。通过定时器老化回收会带来一个问题，如果应用需要维持连接的时间大于NAT网关的设置，通信就会意外中断。因为网关回收相关转换表资源以后，新的数据到达时就找不到相关的转换信息，必须建立新的连接。当这个新数据是由公网侧向私网侧发送时，就会发生无法触发新连接建立，也不能通知到私网侧的主机去重建连接的情况。这时候通信就会中断，不能自动恢复。即使新数据是从私网侧发向公网侧，因为重建的会话表往往使用不同于之前的公网IP和端口地址，公网侧主机也无法对应到之前的通信上，导致用户可感知的连接中断。NAT网关要把回收空闲连接的时间设置到不发生持续的资源流失，又维持大部分连接不被意外中断，是一件比较有难度的事情。在NAT已经普及化的时代，很多应用协议的设计者已经考虑到了这种情况，所以一般会设置一个连接保活的机制，即在一段时间没有数据需要发送时，主动发送一个NAT能感知到而又没有实际数据的保活消息，这么做的主要目的就是重置NAT的会话定时器。

其次，NAT在实现上将多个内部主机发出的连接复用到一个IP上，这就使依赖IP进行主机跟踪的机制都失效了。如网络管理中需要的基于网络流量分析的应用无法跟踪到终端用户与流量的具体行为的关系。基于用户行为的日志分析也变得困难，因为一个IP被很多用户共享，如果存在恶意的用户行为，很难定位到发起连接的那个主机。即便有一些机制提供了在NAT网关上进行连接跟踪的方法，但是把这种变换关系接续起来也困难重重。基于IP的用户授权不再可靠，因为拥有一个IP的不等于一个用户或主机。一个服务器也不能简单把同一IP的访问视作同一主机发起的，不能进行关联。有些服务器设置有连接限制，同一时刻只接纳来自一个IP的有限访问（有时是仅一个访问），这会造成不同用户之间的服务抢占和排队。有时服务器端这样做是出于DoS攻击防护的考虑，因为一个用户正常情况下不应该建立大量的连接请求，过度使用服务资源被理解为攻击行为。但是这在NAT存在时不能简单按照连接数判断。总之，因为NAT隐蔽了通信的一端，把简单的事情复杂化了。

我们来深入理解一下NAT对IP端到端模型的破坏力。NAT通过修改IP首部的信息变换通信的地址。但是在这个转换过程中只能基于一个会话单位。当一个应用需要保持多个双向连接时，麻烦就很大。NAT不能理解多个会话之间的关联性，无法保证转换符合应用需要的规则。当NAT网关拥有多个公有IP地址时，一组关联会话可能被分配到不同的公网地址，这通常是服务器端无法接受的。更为严重的是，当公网侧的主机要主动向私网侧发送数据时，NAT网关没有转换这个连接需要的关联表，这个数据包无法到达私网侧的主机。这些反方向发送数据的连接总有应用协议的约定或在初始建立的会话中进行过协商。但是因为NAT工作在网络层和传输层，无法理解应用层协议的行为，对这些信息是无知的。NAT希望自己对通信双方是透明的，但是在这些情况下这是一种奢望。

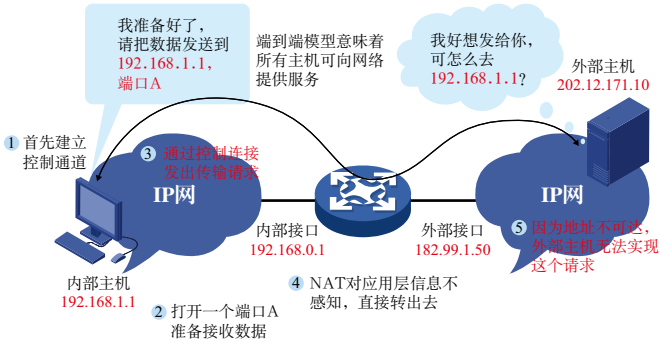


图4 NAT对端到端通信模型的破坏

此外，NAT工作机制依赖于修改IP包头的信息，这会妨碍一些安全协议的工作。因为NAT篡改了IP地址、传输层端口号和校验和，这会导致认证协议彻底不能工作，因为认证目的就是要保证这些信息在传输过程中没有变化。对于一些隧道协议，NAT的存在也导致了额外的问题，因为隧道协议通常用外层地址标识隧道实体，穿过NAT的隧道会有IP复用关系，在另一端需要小心处理。ICMP是一种网络控制协议，它的工作原理也是在两个主机之间传递差错和控制消息，因为IP的对应关系被重新映射，ICMP也要进行复用和解复用处理，很多情况下因为ICMP报文载荷无法提供足够的信

息，解复用会失败。IP分片机制是在信息源端或网络路径上，需要发送的IP报文尺寸大于路径实际能承载最大尺寸时，IP协议层会将一个报文分成多个片断发送，然后在接收端重组这些片断恢复原始报文。IP这样的分片机制会导致传输层的信息只包括在第一个分片中，NAT难以识别后续分片与关联表的对应关系，因此需要特殊处理。

NAT穿越技术

前面解释了NAT的弊端，为了解决IP端到端应用在NAT环境下遇到的问题，网络协议的设计者们创造了各种武器来进行应对。但遗憾的是，这里每一种方法都不完美，还需要在内部主机、应用程序或者NAT网关上增加额外的处理。

应用层网关

应用层网关（ALG）是解决NAT对应用层协议无感知的一个最常用方法，已经被NAT设备厂商广泛采用，成为NAT设备的一个必需功能。因为NAT不感知应用协议，所以有必要额外为每个应用协议定制协议分析功能，这样NAT网关就能理解并支持特定的协议。ALG与NAT形成互动关系，在一个NAT网关检测到新的连接请求时，需要判断是否为已知的应用类型，这通常是基于连接的传输层端口信息来识别的。在识别为已知应用时，再调用相应功能对报文的深层内容进行检查，当发现任何形式表达的IP地址和端口时，将会把这些信息同步转换，并且为这个新连接创建一个附加的转换表项。这样，当报文到达公网侧的目的主机时，应用层协议中携带的信息就是NAT网关提供的地址和端口。一旦公网侧主机开始发送数据或建立连接到此端口，NAT网关就可以根据关联表信息进行转换，再把数据转发到私网侧的主机。很多应用层协议实现不限于一个初始连接（通常为信令或控制通道）加一个数据连接，可能是一个初始连接对应很多后续的新连接。比较特别的协议，在一次协商中会产生一组相关连接，比如RTP/RTCP协议规定，一个RTP通道建立后占用连续的两个端口，一个服务于数据，另一个服务于控制消息。此时，就需要ALG分配连续的端口为应用服务。ALG能成功解决大部分协议的NAT穿越需求，但是这个方法也有很大的限制。因为应用协议的数量非常多而且在不断发展变化之中，添加到设备中的ALG功能都是为特定协议的特定规范版本而开发的，协议的创新和演进要求NAT设备制造商必须跟踪这些协议的最近标准，同时兼容旧标准。尽管有如Linux这种开放平台允许动态加载新的ALG特性，但是管理成本仍然很高，网络维护人员也不能随时了解用户都需要什么应用。因此为每个应用协议开发ALG代码并跟踪最新标准是不可行的，ALG只能解决用户最常用的需求。此外，出于安全性需要，有些应用类型报文从源端发出就已经加密，这种报文在网络中间无法进行分析，所以ALG无能为力。

探针技术STUN和TURN

所谓探针技术，是通过在所有参与通信的实体上安装探测插件，以检测网络中是否存在NAT网关，并对不同NAT模型实施不同穿越方法的一种技术。STUN服务器被部署在公网上，用于接收来自通信实体的探测请求，服务器会记录收到请求的报文地址和端口，并填写到回送的响应报文中。客户端根据接收到的响应消息中记录的地址和端口与本地选择的地址和端口进行比较，就能识别出是否存在NAT网关。如果存在NAT网关，客户端会使用之前的地址和端口向服务器的另外一个IP发起请求，重复前面的探测。然后再比较两次响应返回的结果判断出NAT工作的模式。由前述的一对多转换模型得知，除对称型NAT以外的模型，NAT网关对内部主机地址端口的映射都是相对固定的，所以比较容易实现NAT穿越。而对称型NAT为每个连接提供一个映射，使得转换后的公网地址和端口对不可预测。此时TURN可以与STUN绑定提供穿越NAT的服务，即在公网服务器上提供一个“地址端口对”，所有此“地址端口对”接收到的数据会经由探测建立的连接转发到内网主机上。TURN分配的这个映射“地址端口对”会通过STUN响应发给内部主机，后者将此信息放入建立连接的信令中通知通信的对端。这种探针技术是一种通用方法，不用在NAT设备上为每种应用协议开发功能，相对于ALG方式有一定普遍性。但是TURN中继服务会成为通信瓶颈。而且在客户端中增加探针功能要求每个应用都要增加代码才能支持。

中间件技术

这也是一种通过开发通用方法解决NAT穿越问题的努力。与前者不同之处是，NAT网关是这一解决方案的参与者。与ALG的不同在

于，客户端会参与网关公网映射信息的维护，此时NAT网关只要理解客户端的请求并按照要求去分配转换表，不需要自己去分析客户端的应用层数据。其中UPnP就是这样一种方法。UPnP中文全称为通用即插即用，是一个通用的网络终端与网关的通信协议，具备信息发布和管理控制的能力。其中，网关映射请求可以为客户动态添加映射表项。此时，NAT不再需要理解应用层携带的信息，只转换IP地址和端口信息。而客户端通过控制消息或信令发到公网侧的信息中，直接携带公网映射的IP地址和端口，接收端可以按照此信息建立数据连接。NAT网关在收到数据或连接请求时，按照UPnP建立的表项只转换地址和端口信息，不关心内容，再将数据转发到内网。这种方案需要网关、内部主机和应用程序都支持UPnP技术，且组网允许内部主机和NAT网关之间可以直接交换UPnP信令才能实施。

中继代理技术

准确说它不是NAT穿越技术，而是NAT旁路技术。简单说，就是在NAT网关所在的位置旁边放置一个应用服务器，这个服务器在内部网络和外部公网分别有自己的网络连接。客户端特定的应用产生网络请求时，将定向发送到应用代理服务器。应用代理服务器根据代理协议解析客户端的请求，再从服务器的公网侧发起一个新的请求，把客户端请求的内容中继到外部网络上，返回的相应反方向中继。这项技术和ALG有很大的相似性，它要求为每个应用类型部署中继代理业务，中间服务器要理解这些请求。

特定协议的自穿越技术

在所有方法中最复杂也最可靠的就是自己解决自己的问题。比如IKE和IPsec技术，在设计时就考虑到了如何穿越NAT的问题。因为这个协议是一个自加密的协议并且具有报文防修改的鉴别能力，其他通用方法爱莫能助。因为实际应用的NAT网关基本都是NAPT方式，所有通过传输层协议承载的报文可以顺利通过NAT。IKE和IPsec采用的方案就是用UDP在报文外面再加一层封装，而内部的报文就不再受到影响。IKE中还专门增加了NAT网关是否存在的检查能力以及绕开NAT网关检测IKE协议的方法。

NAT的应用和实现

NAT的应用

NAT在当代Internet中被广泛采用，小至家庭网关，大到企业广域网出口甚至运营商业务网络出口。其实NAT在用户身边随处可见，一般家庭宽带接入的ADSL Modem和SOHO路由器都内置了NAT功能，Windows XP支持网络连接共享，一个用户连接到公网可能会经过多层NAT而对此一无所知。很多企业也为节约IP费用采用NAT接入Internet，但是相比家庭用户有更复杂的需求。

NAT多实例应用

在VPN网络中，多实例路由意味着一个物理拓扑上承载多个逻辑拓扑，网络终端被分配到相互隔离的逻辑拓扑中，彼此之间没有路由的通路。但在访问Internet或者一些关键服务器资源时，被隔离的网络之间又存在共享资源的需求。NAT的多实例实现就是跨越这种逻辑拓扑的方法，把一个空间的网络地址映射到另一个空间。

NAT的高可靠性组网

提高网络可靠性是一个广泛的需求，NAT作为私网到公网的关键路径自然也需要高可靠性。当一个设备提供多个公网接口时，在多接口上部署NAT可以提供更高带宽和多ISP就近访问的能力。但是，当部署多个出口时，访问的流量可能会从不匹配的接口返回，这就要求NAT方案有良好的路由规划和部署合适的策略保证这种流量能够正确处理。在多个物理设备承担NAT功能时，不同设备之间的信息备份和流量分担也是一个组网难题。

同时转换源和目的地址的应用

前面我们介绍的所有NAT应用中，由内网向外网访问过程中，都是将源地址进行转换而目的地址保持不变，报文反方向进入时则处理目的地址。但有一些特殊应用需要在由内向外的IP通路上，替换目的IP地址。通常，这种应用会同时替换源地址和目的地

址，在经过NAT网关以后完成两次地址转换。当两个均规划使用私属IP地址范围的网络进行合并时，终端用户都不想调整自己的IP地址方案，又希望开放一些网络资源给彼此访问。这时就可以通过NAT的两次地址转换来解决路由和地址规划无法解决的问题。

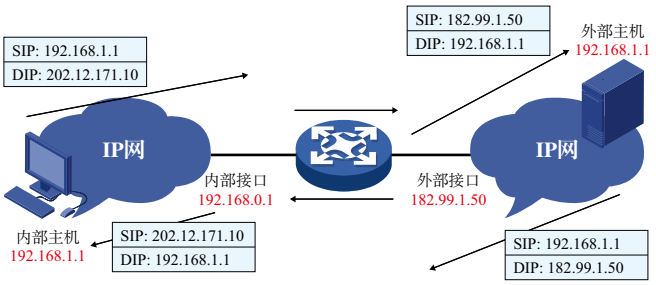


图5 同时转换源和目的地址的应用

NAT的设备实现

NAT作为一个IP层业务特性，在产品实现中与防火墙、会话管理等特性有紧密联系，这是因为NAT判断一个进入设备的报文是否需要NAT处理，判断报文是否为一个新的连接，都需要通过匹配访问控制列表规则和查询会话关联表进行判断。为了满足不同应用场景的NAT需求，NAT的管理界面可提供用户多种配置策略。按照NAT的具体工作方式，又可以做如下分类。

静态一对一地址映射

这种工作方式下，NAT把一个私网地址和一个公网地址做静态关联，在从内而外的方向，将源IP匹配的私网IP替换为公网IP，反方向则将目的IP匹配公网IP的报文替换为私网IP。网络层以上的部分不进行替换处理，只修正校验和。

静态多对多地址映射

这种方式与上一种类似，只是把一段私网地址映射到一段公网地址。工作机制与前述的方式没有差别，只是简化配置工作量。

动态端口映射

这是最基本的工作方式，即前面多次介绍的将一段内网地址动态翻译为一个或多个公网IP，同时对传输层端口或其他上层协议信息进行转换，以实现IP复用。对由内而外的报文，替换源地址和端口，反向报文替换目的地址和端口。仅以连接公网的接口IP作为NAT转换的公网地址时，这种配置最简化，又被称为Easy IP。当以一段公网IP地址作为NAT转换地址时，需要配置一个地址池，NAT会自动在地址池中选择使用公网IP。

动态地址映射（No-PAN）

这是介于静态多对多地址映射和动态端口映射方式之间的一种工作机制。当有一个私网向公网侧访问到达NAT网关时，NAT网关会检查这个私网IP是否已经有关联的公网IP映射。如果已经存在，则按照转换表直接替换IP，不修改上层协议。如果不存在关联表项，则在空闲的公网IP池中占用一个IP，并写入关联表中，以后按照这个关联关系进行地址转换。当这个私网主机发起的所有对外访问均关闭或超时后，回收公网IP。这种方式可以理解为一组内网主机抢占式地共享一个公网IP地址池。当公网IP地址池用完以后，新连接将无法建立。

静态端口映射

通过静态配置，把一个固定的私网IP地址和端口关联到一个公网地址和端口上。这种方式等同于前面介绍过的全锥模式，但是不需要内网主机首先发出报文。这种方式适用于在NAT网关上把一个知名服务（如HTTP）映射到一个内部主机上，也称为port forwarding。

应用层网关（ALG）

在所有NAT产品实现中，ALG是一个必需的功能组件。但在不同实现中，有些产品可以动态加载不同的ALG模块，有些产品可以提供ALG开关控制，有些则不提供任何用户接口。ALG解析上层应用协议的内容，并且根据需要修改IP和端口相关信息，创建和维护附加的关联表项。

NAT转换关联表

无论哪一种NAT工作方式，都要用到地址转换关联表，在不同产品的实现中，这个关联表的存储结构和在IP转发中调用的方式有很大不同。关联表中会记录源IP、目的IP、连接协议类型、传输层源端口、目的端口，以及转换后的源IP、源端口，目的IP、目的端口信息，这里的源和目的都是对应于从内网到外网的访问方向。依据NAT具体工作方式，这些信息可能全部填充，也可能部分填充。例如只按照IP做静态映射的方式，就不需要填入任何端口相关信息；对于静态端口映射，则只填入源相关的内容，而目的端的信息为空。

后IPv4时代的NAT

NAT是为延缓IPv4地址耗尽而推出的技术。毫无疑问，它已经出色完成了自己的历史使命，IPv4比预期走得更远。作为继任者的IPv6吸取了IPv4的教训，被赋予充足地址空间的同时在各个方面做了优化——安全、高效、简洁。但是IPv6无法平滑地取代IPv4，导致IP升级步伐缓慢。尽管网络协议的分层设计很清晰，大量应用层协议和互联网软件中仍内嵌了IPv4地址的处理，要Internet全网升级到IPv6，必须先完成应用的改造。因为NAT和它的穿越技术结合能够满足大部分用户的需求，所以IPv6时代被不断推迟。

随着IPv4地址的濒临耗尽，再经济的模式也无以为继，IPv4必须退出历史舞台。人们自然会认为，NAT作为IPv4的超级补丁技术使命已经完结。实际情况是，IPv4向IPv6过渡的阶段，NAT仍然是一项必不可少的技术手段。因为Internet无法在一日之内完成全网升级，必然是局部升级，逐渐替换。在两套协议并存的时期，用户和服务资源分布在不同网络之间，跨网访问的需求必须得到满足。这正是NAT所擅长的领域，地址替换，因此NAT-PT应运而生。由于IPv4和IPv6之间的差异，NAT要做的事比以往更复杂，有更多的限制和细节。

此外，IETF也在制定纯IPv6网络使用的NAT规范。虽然人们还看不到这种应用的强烈需求，但是NAT仍有其独特的作用，比如隐藏内部网络的地址，实现重叠地址网络的合并等。

毫不夸张地说，正是有了NAT，以IPv4为基础的Internet才能容纳数十亿的用户终端，成就今日之辉煌。IPv4已至日暮西山，IPv6的黎明尚未来临，Internet比任何时刻都更依赖NAT这项过渡技术。NAT的历史再次证明，翻天覆地的划时代进步不一定有市场，抱残守缺的修修补补未必不会成功。在世代更替之时让我们走近NAT，领略IP领域更多细微但不高深的知识，理解NAT就是理解变换万千的应用世界。

NAT转发一些细节的汇总

文/袁亚屏

NAT的基本原理并不复杂，但是在落实到具体的软件实现中，还是有很多的细节值得玩味的。

NAT的基本原理并不复杂，只是在网关设备上维护私网地址（端口）到公网地址（端口）的映射关系，通过修改流经网关的IP报文头部实现地址转换。由于NAT是IP转发流程中的一个环节，具体的实现势必和软硬件密切相关。为了尽可能提高设备转发和报文处理的性能，软件数据结构和处理流程会根据设备硬件的特点进行设计。本节我们就把讨论焦点放在Comware v5平台单核CPU与多核CPU的NAT实现细节上，同时介绍交换机产品支持NAT的方法。

单核还是多核?

由于NAT的业务处理包括对于网络层、传输层甚至应用层的处理，普通的ASIC和交换芯片很难实现，因此一般的NAT实现都是通过CPU或NP来完成的。目前主流的实现方案包括单核CPU、多核CPU和NP三种。以下以Comware v5平台为例，分别介绍在单核CPU硬件平台和多核CPU硬件平台上NAT实现方式的异同点。

单核CPU硬件平台

单核CPU一般用于中低端设备。对于Comware v5平台，NAT的业务处理使用专门的NAT会话来实现，同时针对NAT Server、ASPF、ALG等进行一系列特殊处理。

普通NAT处理

当不涉及ALG等特殊场景时，单核CPU方式的NAT使用NAT会话表来记录会话信息，下面是在PAT或Easy IP方式下，一个从18.1.0.1发起，目的地为18.2.0.1的HTTP连接建立的NAT会话：

Protocol	GlobalAddr	Port	InsideAddr	Port	DestAddr	Port
TCP	18.2.255.254	12288	18.1.0.1	1912	18.2.0.1	80
status：111	TTL：00:00:10		Left：00:00:06		VPN：---	

同样一个连接，如果使用No-PAT方式的NAT，会话信息如下：

Protocol	GlobalAddr	Port	InsideAddr	Port	DestAddr	Port
-	18.2.1.1	---	18.1.0.1	---	---	---
status：NOPAT	TTL：00:04:00		Left：00:04:00		VPN：---	
TCP	18.2.1.1	1934	18.1.0.1	1934	18.2.0.1	80
status：NOPAT	TTL：00:00:10		Left：00:00:04		VPN：---	

No-PAT方式下的会话其实是由一个主会话和多个子会话组成的。在该方式下，NAT模块依旧要根据子会话对端口进行判断，确定返回的报文的端口和发出的报文匹配，才进行NAT转换，防止满足第一条session条件的报文（仅需要二元组匹配）从外网轻松穿越NAT。这样在No PAT的转换方式下，同样保证了只能由内网主动发起连接，提供了更可靠的安全性。

ALG的处理

多数协议在经过NAT网关时只需要使用普通的NAT转换处理，但是对于一些特殊协议如FTP、RTSP、SIP等，由于这些协议在应用层中携带了IP地址或端口等信息，因此需要ALG的帮助才能正常穿越NAT网关。

以FTP协议为例，PORT模式的FTP在经过NAT时需要ALG的处理，因此NAT网关需要关注用户的每一个FTP命令，并识别需要进行ALG处理的命令，进行相应的ALG处理：

```
* May 28 10:54:30:820 2011 MSR50 NAT/7/debug:
  ftp Packet To Svr（Vlan-interface102-out：）Find a ftp PORT CMD
* May 28 10:54:30:821 2011 MSR50 NAT/7/debug:
  ftp Packet To Svr（Vlan-interface102-out：）Process the PORT command 18.1.0.1:1976
```

以上内容是通过打开设备调试开关跟踪到的信息。由此可以看出，NAT网关识别了FTP PORT命令，并进行了ALG的处理，那么究竟进行了哪些ALG的处理呢?

当数据通道连接建立完毕后，NAT的会话表如下：

Protocol	GlobalAddr	Port	InsideAddr	Port	DestAddr	Port
-	18.2.1.1	---	18.1.0.1	---	---	---
status：NOPAT	TTL：00:04:00		Left：00:04:00		VPN：---	
TCP	18.2.1.1	1998	18.1.0.1	1998	18.2.0.1	21
status：NOPAT	TTL：00:05:00		Left：00:04:57		VPN：---	
TCP	18.2.1.1	2002	18.1.0.1	2002	18.2.0.1	20
status：NOPAT	TTL：00:05:00		Left：00:04:57		VPN：---	
TCP	18.2.1.1	2002	18.1.0.1	2002	---	---
status：NOPAT	TTL：---		Left：---		VPN：---	

此处关注最后一个标红的会话，其目的地址和目的端口都为空，表示任何一个外网的IP地址都可以使用任意源端口来连接这个2002端口。这个会话正是为了适应FTP的“第三方连接”特性而设计的，是NAT ALG工作的结果。

从上面内容可以看出，对于单核CPU的NAT，ALG工作的效果是直接体现在NAT会话中的。

和ASPF的共存

ASPF（Application Specific Packet Filter）是一种基于应用层状态的包过滤方法。ASPF能够检查应用层协议信息，如报文的协议类型和端口号等信息，并且监控基于连接的应用层协议状态。每一个连接状态信息都将被ASPF维护，并用于动态地决定数据包是否被允许通过防火墙进入内部网络，以阻止恶意的入侵。 同时，ASPF能够检测传输层协议信息（即TCP/UDP检测），根据源、目的IP地址及端口号决定TCP或UDP报文是否可以通过防火墙进入内部网络。

对于单核产品，当在设备上配置了应用层协议检测后，ASPF可以检测每一个应用层的会话，并创建一个状态表项和一个临时访问控制列表（Temporary Access Control List，TACL），对于多通道协议，后续还会创建数据通道的TACL：

■ 状态表项在ASPF检测到第一个向外发送的报文时创建，用于维护一次会话中某一时刻会话所处的状态，并检测会话状态的转换是否正确；

■ 临时访问控制列表的表项在创建状态表项的同时创建，会话结束后删除，它相当于一个扩展ACL的Permit项。TACL主要用于匹配一个会话中的所有返回的报文，为某一应用返回的报文在防火墙的外部接口上建立一个临时的返回通道。

ASPF一般用于阻止外网发起到内网的连接，对于普通的内网访问外网的NAT并没有影响，但是对于PORT模式的FTP，由于数据连接是从外部服务器发起的，因此还是存在影响的。

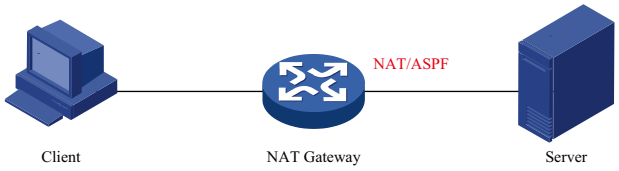


图1 NAT/ASPF并存

如上图，NAT和ASPF并存，当使用PORT模式的FTP时，ASPF实际上进行了特殊处理，放过了入方向的数据连接：

```
*May 28 11:29:10:620 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 33，token = 32，string 'PORT'，value 3

*May 28 11:29:10:620 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 33，token = 44，string '8'，value 8

*May 28 11:29:10:620 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 33，token = 13，string '77'，value 77

//捕获了PORT命令和命令中的端口

*May 28 11:29:10:620 2011 MSR50 ASPF/7/EVENT: Source address :18.2.0.1 port number:（1:65535）destination address :18.1.0.1 port number:（2125:2125）

*May 28 11:29:10:621 2011 MSR50 ASPF/7/OBJ_CREATE: Create session entry 0xB36BA40 address 18.2.0.1 bucket 2131

//为服务器的地址建立ASPF会话

*May 28 11:29:10:621 2011 MSR50 ASPF/7/OBJ_CREATE: Create the temporary ACL entry 0x7D67D00 originator 18.2.0.1（1:65535）destination 18.1.0.1（2125:2125）bucket 2131

//为入方向数据连接建立TACL

*May 28 11:29:10:621 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 34，token = 32，string '200'，value 3

*May 28 11:29:10:621 2011 MSR50 ASPF/7/ftp: session 0xB36BA40 - process the PORT command 18.2.0.1:0

*May 28 11:29:10:623 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 33，token = 32，string 'LIST'，value 255

*May 28 11:29:10:773 2011 MSR50 ASPF/7/ftp: session 0xB36CE80 - state = 33，token = 32，string '150'，value 254

*May 28 11:29:10:873 2011 MSR50 ASPF/7/EVENT: Source address :18.2.0.1 port number:（20:20）destination address :18.1.0.1 port number:（2125:2125）

*May 28 11:29:10:973 2011 MSR50 ASPF/7/OBJ_CREATE: Create the temporary ACL entry 0xB5CEAA0 originator 18.1.0.1（2125:2125）destination 18.2.0.1（20:20）bucket 2149

//为出方向数据连接建立TACL

*May 28 11:29:11:123 2011 MSR50 ASPF/7/OBJ_DELETE: Delete the temporary ACL entry 0xB5CEAA0 originator 18.1.0.1（2125:2125）destination 18.2.0.1（20:20）bucket 2149

*May 28 11:29:11:274 2011 MSR50 ASPF/7/OBJ_DELETE: Delete session entry 0xB36BA40 address 18.2.0.1 bucket 2151

//删除入方向TACL和会话

*May 28 11:29:11:424 2011 MSR50 ASPF/7/OBJ_DELETE: Delete the temporary ACL entry 0x7D67D00 originator 18.2.0.1（20:20）destination 18.1.0.1（2125:2125）bucket 2151

//删除出方向TACL
```

NAT Server

NAT Server实际上是反方向的NAT，需要用户进行静态配置。需要注意的是，对于单核CPU设备，经过NAT Server的流量不会再另外创建NAT会话。

对于NAT Server来说，如果和ASPF共存，需要在ASPF的入方向包过滤引用的ACL中将NAT Server对应的IP/端口放开，如下图：

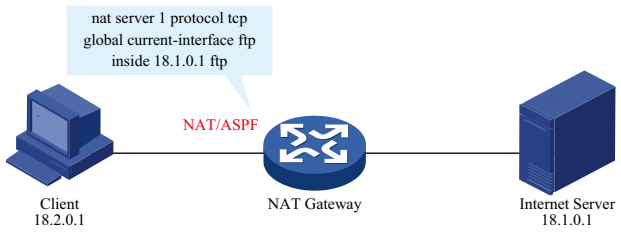


图2 NAT Server和ASPF

在接口的入方向，ASPF是先于NAT对报文进行处理的，因此ASPF的入方向包过滤引用的ACL应进行如下的配置，NAT Server才能正常工作：

```
#
acl number 3001
rule 0 permit icmp
rule 1 permit tcp destination 18.1.0.1 0 destination-port eq ftp
rule 5 deny ip
#
```

同样，NAT Server对于特殊的应用层协议也需要ALG的支持，对于Passive模式的FTP客户端，在访问NAT Server时需要ALG处理：

```
* May 28 12:19:06:505 2011 MSR50 NAT/7/debug:
  ftp Packet To Client （Vlan-interface102-out：）Find a ftp Entering PASV CMD
* May 28 12:19:06:605 2011 MSR50 NAT/7/debug:
  ftp Packet To Client （Vlan-interface102-out：） Process the PASV command
  18.1.0.1:2330
```

回过头来看NAT会话：

Protocol	GlobalAddr	Port	InsideAddr	Port	DestAddr	Port
TCP	18.2.255.254	2343	18.1.0.1	2343	18.2.0.1	3590
status：241	TTL：00:05:00		Left：00:03:48		VPN：---	
TCP	18.2.255.254	2343	18.1.0.1	2343	---	---
status：8241	TTL：---		Left：---		VPN：---	

同样，最下面一条会话也是为了支持FTP的“第三方连接”功能。

那么，当NAT Server/ASPF/ALG共存，会出现哪些问题呢？

如图2，当接口配置如下时，Passive模式的FTP客户端，数据连接建立失败：

```
#
interface Vlan-interface102
ip address 18.2.255.254 255.255.0.0
NAT outbound 2000
NAT Server 1 protocol tcp global current-interface ftp inside 18.1.0.1 ftp
firewall packet-filter 3001 inbound
firewall aspf 1 outbound
#
```

FTP数据连接是从客户端发起的TCP连接，因此被入方向的packet filter过滤掉了，只要在ACL3001中增加对应的配置即可。但是对于Passive模式的FTP数据连接来说，源端口和目的端口都不是固定的，无法进行ACL的配置。

解决问题的方法是另外配置一条入方向的ASPF：

```
#
interface Vlan-interface102
ip address 18.2.255.254 255.255.0.0
NAT outbound 2000
NAT Server 1 protocol tcp global current-interface ftp inside 18.1.0.1 ftp
firewall packet-filter 3001 inbound
firewall aspf 1 inbound
firewall aspf 1 outbound
#
```

入方向的ASPF对FTP协议进行了检测，并放行了数据连接：

```
*May 28 12:40:05:171 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 33,
token = 32, string 'PASV', value 4
//检测到PASV命令
```

```
*May 28 12:40:05:321 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 35, token = 32, string '227', value 4

*May 28 12:40:05:422 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 35, token = 44, string '9', value 9

*May 28 12:40:05:522 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 35, token = 41, string '115', value 115

*May 28 12:40:05:672 2011 MSR50 ASPF/7/EVENT: Source address :18.2.0.1 port number: ( 1:65535 ) destination address :18.1.0.1 port number: ( 2419:2419 )

*May 28 12:40:05:772 2011 MSR50 ASPF/7/OBJ_CREATE: Create session entry 0xB346A80 address 18.2.0.1 bucket 2425

*May 28 12:40:05:922 2011 MSR50 ASPF/7/ftp: session 0xB346A80 - process the reply to the PASV command 18.1.0.1:2419

//开始处理对PASV命令的应答消息

*May 28 12:40:06:022 2011 MSR50 ASPF/7/OBJ_CREATE: Create the temporary ACL entry 0x9226360 originator 18.2.0.1 ( 1:65535 ) destination 18.1.0.1 ( 2419:2419 ) bucket 2425

//针对数据连接的端口创建入方向TACL

*May 28 12:40:06:122 2011 MSR50 ASPF/7/EVENT: Source address :18.2.0.1 port number: ( 3621:3621 ) destination address :18.1.0.1 port number: ( 2419:2419 )

*May 28 12:40:06:273 2011 MSR50 ASPF/7/OBJ_CREATE: Create the temporary ACL entry 0x9226420 originator 18.1.0.1 ( 2419:2419 ) destination 18.2.0.1 ( 3621:3621 ) bucket 6044

//针对数据连接的端口创建出方向TACL

*May 28 12:40:06:373 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 33, token = 32, string 'LIST', value 255

*May 28 12:40:06:473 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 33, token = 32, string '125', value 254

*May 28 12:40:06:623 2011 MSR50 ASPF/7/ftp: session 0xB346CC0 - state = 33, token = 32, string '226', value 254

*May 28 12:40:09:531 2011 MSR50 ASPF/7/OBJ_DELETE: Delete the temporary ACL entry 0x9226420 originator 18.1.0.1 ( 2419:2419 ) destination 18.2.0.1 ( 3621:3621 ) bucket 6044

//传输结束后删除出方向TACL

*May 28 12:40:09:531 2011 MSR50 ASPF/7/OBJ_DELETE: Delete session entry 0xB346A80 address 18.2.0.1 bucket 6046

*May 28 12:40:09:531 2011 MSR50 ASPF/7/OBJ_DELETE: Delete the temporary ACL entry 0x9226360 originator 18.2.0.1 ( 3621:3621 ) destination 18.1.0.1 ( 2419:2419 ) bucket 6046

//传输结束后删除入方向TACL
```

静态NAT

单核CPU设备支持一对一和网段对网段的静态NAT，被静态NAT绑定的内网地址可以通过对应的外网地址从外网直接进行访问，对于ALG的需求和普通的NAT Outbound以及NAT Server并没有区别。

静态NAT处理的流量要创建NAT会话，会话有些类似No-PAT的NAT：

Protocol	GlobalAddr	Port	InsideAddr	Port	DestAddr	Port
-	18.2.1.1	0	18.1.0.1	0	---	---
status：800	TTL：00:05:00		Left：00:04:57		VPN：---	
TCP	18.2.1.1	3477	18.1.0.1	3477	18.2.0.1	21
status：60008	TTL：00:05:00		Left：00:04:58		VPN：---	

需要注意的是，静态NAT由于能够保证地址一一对应，因此不再做端口的转换。

多核CPU硬件平台

多核CPU被视为一种有效提升软转发性能的手段，使用范围越来越广。Comware v5的NAT功能针对多核硬件平台的实现和针对单核硬件平台的实现有很大的不同。

会话管理

在Comware v5平台中，会话管理是多核CPU设备所特有的功能。会话管理是为了实现NAT、ASPF、攻击检测及防范等基于会话进行处理的业务而抽象出来的公共模块。会话管理把传输层报文之间的交互关系抽象为会话，并根据发起方或响应方的报文信息对会话进行状态更新和超时老化。

会话管理支持多个特性分别对同一个报文进行处理，实现的主要功能包括：

- 报文到会话的快速匹配；
- 传输层协议状态的管理；
- 报文应用层协议类型的识别；
- 支持会话按照协议状态或应用层协议类型进行老化；
- 支持指定会话维持永久连接；
- 会话的传输层协议报文校验和检查；
- 为需要进行端口协商的应用层协议提供特殊的报文匹配；

■ 支持对ICMP差错控制报文的解析以及根据解析结果进行会话的匹配。

对于NAT来说，在多核CPU设备上没有独享的会话表，而是和ASPF等协议使用共同的会话表。这种实现大大简化了多种应用在一个接口上并存情况下的互操作。

经NAT处理的普通流量创建的会话表如下：

```
Initiator:
Source IP/Port：18.1.0.1/2545
Dest IP/Port：18.2.0.1/80
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port：18.2.0.1/80
Dest IP/Port：18.2.255.254/1026
VPN-Instance/VLAN ID/VLL ID:
Pro：TCP（6）        App：HTTP        State：TCP-EST
Start time：2011-05-28 13:07:25  TTL：3587s
Received packet（s）（Init）：27 packet（s）    1465 byte（s）
Received packet（s）（Reply）：36 packet（s）    48751 byte（s）
```

上表中已经明确列出了流量发起方（Initiator）的源、目的地址和响应方（Responder）的源、目的地址，因为经过了NAT的处理，所以二者并不完全对称。会话表比单核CPU设备的NAT会话表包含更多的信息，也更为直观。

ALG的处理

提到多核CPU设备的ALG，不得不谈的就是关联表。

对于单核CPU设备来讲，ALG处理无法共享代码，同样一个FTP ALG，NAT需要对其进行特殊处理，ASPF也要对其进行特殊处理，大家各自维护一套代码，效率低下且不利于维护管理。

正如会话表统领了NAT会话表、ASPF会话表等表项一样，关联表被用来处理一切和ALG有关的问题。

和“单核CPU硬件处理平台ALG的处理”一节同样的场景，NAT网关为多核CPU设备，当ALG检测到客户端的PORT命令时，会创建关联表：

```
*May 28 13:18:38:400 2011 SR66 ALG/7/ALG_DBG: -Slot=2; Alg debug info:
Receive a ftp PORT CMD packet
*May 28 13:18:38:400 2011 SR66 ALG/7/ALG_DBG: -Slot=2; Alg debug info:
From VPN: 0    Pro: TCP
Direction : OUT
(18.1.0.1: 2616) ----> (18.2.255.254: 1039)

*May 28 13:18:38:454 2011 SR66 session/7/RELATION: -Slot=2;
LocalTuple3: 18.1.0.1/2616 : GlobalTuple3:18.2.255.254/1039 (TCP)
Create module call
```

关联表信息如下：

Local IP/Port	Global IP/Port		MatchMode
18.1.0.1/2616	18.2.255.254/1039		Global
App：ftp-data	Pro：TCP	TTL：300s	AllowConn 1

关联表中各字段的含义如下：

字段	描述
Local IP/Port	内网IP地址/端口号
Global IP/Port	外网IP地址/端口号
MatchMode	会话表向关联表匹配模式，包括：Local、Global、Either ■ Local表示新建会话的源IP/源端口与关联表的Local IP/Port匹配 ■ Global表示新建会话的目的IP/目的端口与关联表的Global IP/Port匹配 ■ Either表示新建会话的信息与关联表的Local IP/Port或Global IP/Port匹配
App	应用层协议类型，包括：FTP、MSN、QQ等
Pro	传输层协议类型，包括：TCP、UDP
TTL	关联表的剩余存活时间，单位为秒
AllowConn	关联表允许创建的会话数

表1 关联表中各字段的含义说明

上面的关联表被用来匹配服务器发起的FTP数据连接，对于FTP协议来说，一个PORT命令只能对应一个数据连接，因此关联表中“AllowConn”字段的值为1，当数据连接建立成功后，关联表的连接数满，被删除：

```
* May 28 13:18:38:464 2011 SR66 session/7/RELATION: -Slot=2;
LocalTuple3: 18.1.0.1/2616 : GlobalTuple3:18.2.255.254/1039 (TCP)
Delete child full
```

```
* May 28 13:18:38:464 2011 SR66 session/7/RELATION：-Slot = 2；
LocalTuple3: 18.1.0.1/2616 : GlobalTuple3:18.2.255.254/1039（TCP）
Delete  module call / child full
```

总之，对于多核CPU的NAT，ALG主要通过关联表进行工作。

和ASPF的共存

多核CPU设备的ASPF和单核CPU设备不同，不存在TACL的概念。ASPF的应用层协议检测功能由会话管理及ALG功能协作实现。ASPF将检测到的应用层会话的首报文与配置的策略进行匹配，匹配的结果交由会话管理用来建立会话信息数据库以及维护会话状态。之后，ASPF根据会话管理返回的会话状态信息决定后续报文的处理方式。

因此，一般来说多核CPU设备的NAT和ASPF之间是比较松散的耦合关系，二者通过会话管理联系。

NAT Server

对于多核CPU设备，经过NAT Server的流量还是会创建会话，这一点与单核CPU处理不同。NAT Server对于特殊的应用层协议也需要ALG的支持，对于Passive模式的FTP客户端，在访问NAT Server时需要ALG处理：

```
*May 28 14:05:06:350 2011 SR66 ALG/7/ALG_DBG: -Slot = 2； Alg debug info：
Receive a ftp PASV Response packet
*May 28 14:05:06:350 2011 SR66 ALG/7/ALG_DBG: -Slot = 2； Alg debug info：
From VPN ： 0     Pro：TCP
Direction：OUT
（18.1.0.1：2793）---->（18.2.255.254：2793）
```

静态NAT

和单核CPU设备一样，多核CPU设备也支持一对一和网段对网段的静态NAT，被静态NAT绑定的内网地址可以通过对应的外网地址从外网直接进行访问，对于ALG的需求和普通的NAT Outbound以及NAT Server并没有区别。

静态NAT处理的流量要创建会话表。

单核架构和多核架构实现的区别

从上面描述可见，单核架构的NAT实现和多核有很大的区别，主要表现在以下几个方面：

	会话表管理	ALG处理	NAT Server
单核构架	使用独立的NAT会话表	处理结果在NAT会话中体现	经过NAT Server流量不创建会话
多核构架	多个业务使用统一的会话表	处理结果用关联表体现	经过NAT Server流量创建会话

表2 单核NAT和多核NAT的区别

NAT插卡相关的转发实现

对于中高端交换机产品来说，NAT功能需要专门的业务插卡来实现，如何将流量引入和引出插卡是转发流程的关键所在。

二三层转发方式

这种方式是将NAT插卡视为一台独立的设备，交换机只做二层转发，报文在入VLAN A被转发到插卡的三层接口A，经插卡内部的三层转发到达三层接口B进行NAT处理，处理完毕后转发到交换机的VLAN B，最终二层转发到外网，外网返回的报文走相反的流程，如下图：

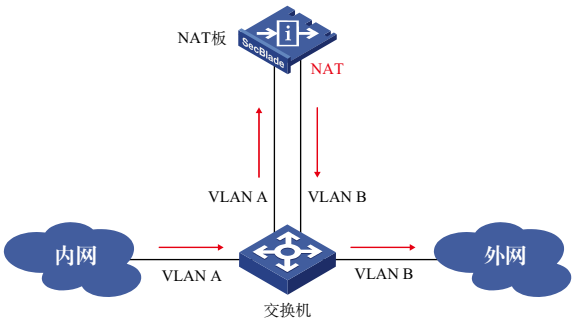


图3 二三层转发方式引流

这种引流方式的原理和配置都比较简单，一般用于交换机和SecBlade插卡配合的场景，但是缺点也很明显，引流难以控制，所有流量都上插卡可能导致插卡负担过重，成为性能瓶颈。

重定向方式

第二种引流的方式是通过重定向的方式将需要NAT处理的流量送上插卡进行处理，不需要NAT处理的流量由交换机直接转发。

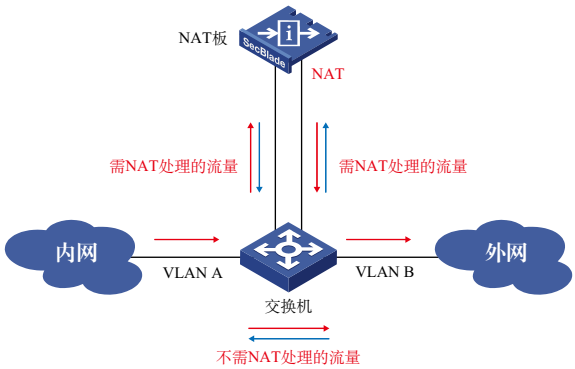


图4 重定向方式引流

这种引流方式克服了方法一的缺点，重定向方式的引流可以有效的控制上NAT插卡的流量，达到优化网络性能的目的。但是这种引流方式在配置方面还是比较繁琐，需要在NAT插卡和交换机两方面分别进行NAT的配置和引流的配置，由于引流配置的固定性，难以实现多块NAT板的备份功能。

OAA方式

OAA方式的引流是NAT插卡通过ACFP协议获取交换机的端口信息，然后通过ACFP协议下发策略，将特定端口的特定流量引到插卡进行NAT处理，处理完毕送回交换机进行转发。

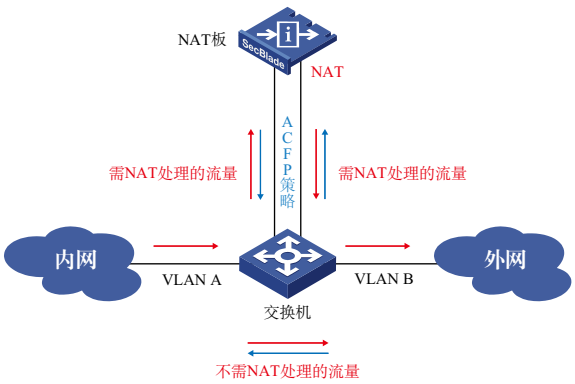


图5 OAA方式引流

OAA方式引流的配置简单，大部分配置都在插卡上完成，交换机只需使能ACFP协议，另外，多块NAT插卡还可以下发相同规则但优先级不同的ACFP策略，实现NAT插卡之间的备份。

小结

对于不同的硬件平台，转发方面的设计和实现必然会有所区别。针对单核CPU和多核CPU两种硬件构架，Comware v5平台设计了两种NAT实现，目的是为了更好的适配硬件平台，实现功能、性能和可维护性等方面的最优化。

丰富的业务插卡是H3C中高端交换机的亮点之一，而多样化的引流方式，尤其是OAA的引流方式增强了NAT插卡的灵活性。

NAT的特殊处理

文/曾劼

在全球IPv4地址愈发匮乏的大背景下，NAT技术应运而生，并且随着时间的推移，这项技术运用的越来越广泛。在实际应用中，NAT大体可以分成Easy IP、PAT、No PAT、静态NAT和NAT Server几种用法。

NAT技术的原理并不复杂，如图1所示，三个带有内部地址的数据报文到达NAT设备，其中报文1和报文2来自同一个内部地址但有不同的源端口号，报文1和报文3来自不同的内部地址但具有相同的源端口号。通过NAT映射，三个数据报的源IP地址都被转换到同一个外部地址，但每个数据报文都被赋予了不同的源端口号，因而仍保留了报文之间的区别。当各报文的回应报文到达时，NAT设备仍能够根据回应报文的目的IP地址和目的端口号来区别该报文应转发到的内部主机。

Direction	Before NAT	After NAT
Outbound	192.168.1.2:1111	20.1.1.1:1001
Outbound	192.168.1.2:2222	20.1.1.1:1002
Outbound	192.168.1.3:1111	20.1.1.1:1003

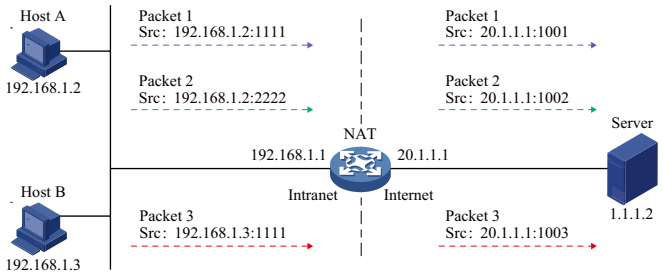


图1 源地址转换原理图

NAT设备通过建立五元组（源地址、源端口号、协议类型、目的地址、目的端口号）表项为依据进行地址分配和报文过滤。即，对于来自相同源地址和源端口号的报文，若其目的地址和目的端口号不同，通过映射后，相同的源地址和源端口号将被转换为不同的外部地址和端口号，并且NAT设备只允许这些目的地址对应的外部网络的主机才可以通过该转换后的地址和端口来访问这些内部网络的主机。

原理虽然简单，但是在实际应用中还是会碰到各种的问题，比如ICMP报文没有端口号，设备如何进行NAT映射？分片报文中没有四层信息，设备又如何进行NAT映射？本文将以H3C SecPath F1000-E设备为例，对NAT的特殊处理进行深入的解析。

地址统一管理

首先我们在设备上配置一个PAT方式的NAT转换，一个No PAT方式的NAT转换，再配置两个NAT Server：

```
#
NAT address-group 3 218.197.70.10 218.197.70.19
NAT address-group 13 61.154.70.20 61.154.70.29
#
interface GigabitEthernet0/3
port link-mode route
NAT outbound 3001 address-group 13 no-pat
NAT outbound 3000 address-group 3
NAT Server protocol tcp global 218.197.70.20 www inside 192.168.200.2 www
NAT Server protocol udp global 218.197.70.21 tftp inside 192.168.200.2 tftp
ip address 218.197.70.1 255.255.255.0
#
```

当我们查看路由表的时候可以看到NAT地址池的地址以及NAT Server的公网地址全部被加入其中。

```
<SecPath F1000-E>display ip routing-table
Routing Tables: Public
Destinations : 31        Routes : 31

Destination/Mask    Proto   Pre  Cost      NextHop         nterface
0.0.0.0/0           Static  60   0         162.105.12.1    GE0/0
```

61.154.70.20/32	Static	1	0	0.0.0.0	NULL0
61.154.70.21/32	Static	1	0	0.0.0.0	NULL0
61.154.70.22/32	Static	1	0	0.0.0.0	NULL0
61.154.70.23/32	Static	1	0	0.0.0.0	NULL0
61.154.70.24/32	Static	1	0	0.0.0.0	NULL0
61.154.70.25/32	Static	1	0	0.0.0.0	NULL0
61.154.70.26/32	Static	1	0	0.0.0.0	NULL0
61.154.70.27/32	Static	1	0	0.0.0.0	NULL0
61.154.70.28/32	Static	1	0	0.0.0.0	NULL0
61.154.70.29/32	Static	1	0	0.0.0.0	NULL0
127.0.0.0/8	Direct	0	0	127.0.0.1	InLoop0
127.0.0.1/32	Direct	0	0	127.0.0.1	InLoop0
162.105.12.0/24	Direct	0	0	162.105.12.211	GE0/0
162.105.12.211/32	Direct	0	0	127.0.0.1	InLoop0
192.168.200.0/24	Direct	0	0	192.168.200.1	GE0/0
192.168.200.1/32	Direct	0	0	127.0.0.1	InLoop0
218.197.70.0/24	Direct	0	0	218.197.70.1	GE0/0
218.197.70.1/32	Direct	0	0	127.0.0.1	InLoop0
218.197.70.10/32	Static	1	0	0.0.0.0	NULL0
218.197.70.11/32	Static	1	0	0.0.0.0	NULL0
218.197.70.12/32	Static	1	0	0.0.0.0	NULL0
218.197.70.13/32	Static	1	0	0.0.0.0	NULL0
218.197.70.14/32	Static	1	0	0.0.0.0	NULL0
218.197.70.15/32	Static	1	0	0.0.0.0	NULL0
218.197.70.16/32	Static	1	0	0.0.0.0	NULL0
218.197.70.17/32	Static	1	0	0.0.0.0	NULL0
218.197.70.18/32	Static	1	0	0.0.0.0	NULL0
218.197.70.19/32	Static	1	0	0.0.0.0	NULL0
218.197.70.20/32	Static	1	0	0.0.0.0	NULL0
218.197.70.21/32	Static	1	0	0.0.0.0	NULL0

如图2所示，从内网来的报文经过NAT转换，到达外网中的目的地，目的主机向NAT转换后的地址回应报文；当报文到达与NAT出接口直连的设备时，这个设备并不知道这个经过转换的公网地址所对应的MAC，此时便会发送ARP请求。

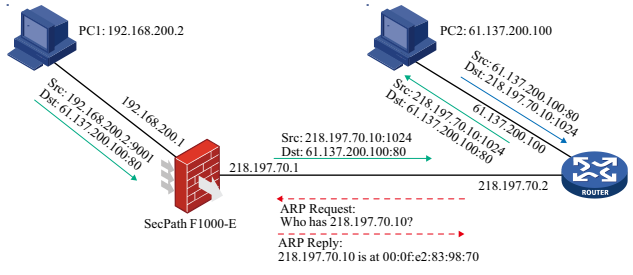


图2 NAT设备的ARP响应

NAT使用的公网地址，包括了被引用的地址池的地址、NAT Server的公网地址和被引用的静态配置的公网地址。这些地址全部会进入地址管理模块进行统一地址管理，并由地址管理模块通知路由管理模块将这些IP地址加入路由表。当外部设备向这些地址发起ARP请求，设备便会将ARP请求送到地址管理模块检查ARP报文的合法性，地址管理发现ARP所请求的IP地址为NAT地址池地址，则通知NAT模块判断是否需要回应ARP，NAT模块会检查ARP所请求的IP地址是否是收到ARP接口下配置的NAT的地址，如果是则通知ARP模块进行ARP应答。

由于有了地址统一管理，与接口不在同一网段的地址池的地址被加入本地路由表，并且可以有选择的将其引入动态路由协议，通过OSPF或者BGP将其发布出去，简化了组网中其他设备的配置。

地址池的优先级

双机热备对于防火墙来说是一个必不可少的功能，两台防火墙上的NAT配置也需要完全相同，这样就会出现一个问题：如果两个防火墙分别将两条不同的流映射到相同的公网地址，并且端口也相同的话，势必会造成表项的混乱，所以我们引入了地址池优先级的概念。

在双机热备的环境中，如果地址池被配置为高优先级，在进行端口映射的时候端口取值范围为1024~35000；如果地址池被配置为低优先级，其端口取值范围为35001~65535。这样主备两台防火墙虽然使用相同的NAT地址池中的地址，但是由于地址池的优先级不同，所以就不会出现NAT转换后公网IP和公网端口完全相同的情况了。

地址池优先级只有在双机热备的环境中才有意义，如果是单机环境，高优先级的地址池的端口取值范围就是1024~65535，而低优先级地址池的端口取值范围依旧是35001~65535。

PING操作的NAT映射

H3C Comware v5平台为了保证内网的安全性，采用了五元组匹配NAT映射表项，但是ICMP报文没有类似于TCP和UDP报文的端口

信息，既要对ICMP的请求报文进行NAT映射，又要确保网络的安全性，只允许合法的ICMP响应报文进入内网，就必须对其进行特殊的处理。

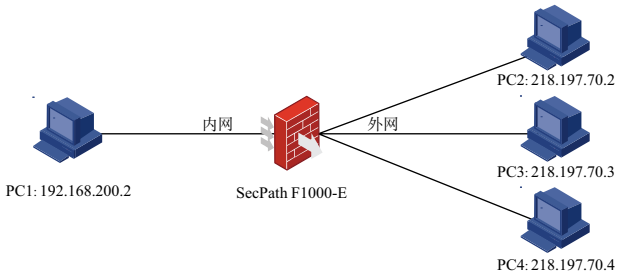


图3 NAT简单组网图

图3是一个最简单的NAT组网图，内网使用192.168.200.0/24的私网地址，外网是218.197.70.0/24的公网地址，而在设备连接外网的接口上我们启用了NAT。我们从内网的PC1去PING外网的PC2是可以成功的，而我们在设备上看到了这样的一个表项：

Initiator:
Source IP/Port : 192.168.200.2/2048
Dest IP/Port : 218.197.70.2/1024
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port : 218.197.70.2/0
Dest IP/Port : 218.197.70.12/1036
VPN-Instance/VLAN ID/VLL ID:
Pro: ICMP（1） App: unknown State: ICMP-CLOSED
Start time: 2011-05-28 11:44:35 TTL: 23s
Root Zone（in）: Private
Zone（out）: Public
Received packet（s）（Init）: 4 packet（s） 240 byte（s）
Received packet（s）（Reply）: 4 packet（s） 240 byte（s）

既然ICMP报文没有端口信息，那么表项中的端口信息又是从何而来的呢？我们不妨通过对报文的分析找到答案。

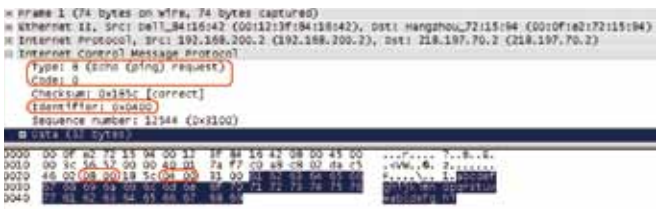


图4 内网ICMP请求报文

从内网抓到的ICMP请求报文可以看出TYPE + CODE字段的值是0×0800，而Identifier字段的值是0×0400，这两个值转换成十进制的数值，正好是2048和1024。与发起方的源端口数值和目的端口数值一致。那么我们再看一下内网抓到的ICMP响应报文是否可以与响应方的源端口和目的端口对应。

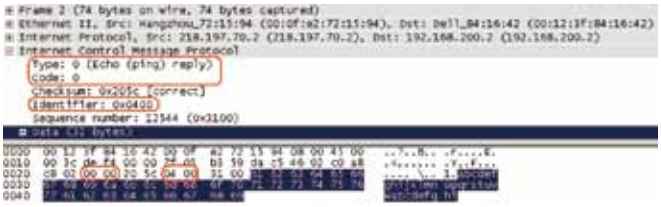


图5 内网ICMP响应报文

在响应报文中TYPE + CODE字段为0×0000，十进制也是0，这与表项中响应方的源端口相同；但是问题出现在响应方的目的端口，在报文中Identifier字段的值是0×0400，十进制为1024，而表项中的目的端口却为1036，这又是为什么呢？原来Windows系统发出的ICMP报文的Identifier字段全部都为0×0400，当多台内网PC同时PING一个外网PC时，如果设备仅仅对Identifier字段进行简单的运算，极有可能造成多个表项中响应方的参数相同，使得回程报文发送错误。为避免这种情况的发生，设备会在对ICMP请求报文进行IP地址转换的同时，还会选择一个空闲的端口号对报文中的Identifier字段进行变化，以避免上述情况的发生。

图6 外网ICMP请求报文

在图6中我们可以看到转换后的ICMP请求报文的Identifier字段已经变成了0×040c，也就是十进制的1036，这与表项中响应方的目的端口一致，这样的一个五元组就能唯一确定一个PING操作了。

ICMP差错报文的处理

内网PC1试图通过TFTP协议从外网的PC2下载aaa.bin文件，但是PC2并没有开启TFTP服务，这时PC2会向PC1回应ICMP端口不可达的差错报文。如果我们依然按照上一节中的规则来理解ICMP差错报文的转换过程，那就大错特错了。对于源地址转换的NAT来说，外网的报文要想顺利进入内网，就必须五元组匹配设备上的NAT表项，内网访问外网时使用的是UDP协议，而ICMP差错报文是ICMP协议，这一点就不符合要求。但是从图7内网PC1上抓包我们清楚的看到这个ICMP差错报文确确实实的被转发了进来。这其中的奥妙就在于设备对ICMP差错报文的特殊处理。

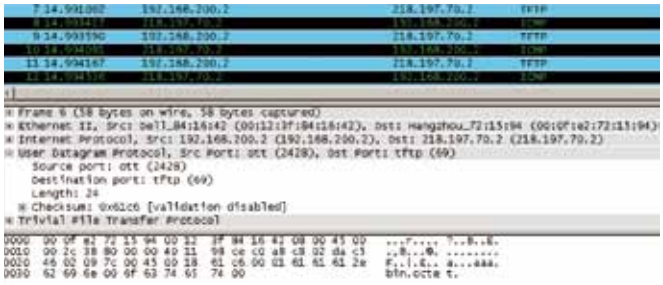


图7 内网TFTP访问抓包

当PC1访问PC2时，设备上会生成这样一个表项，PC1使用的是UDP协议，源IP和源端口为192.168.200.2/2428，目的IP和目的端口为218.197.70.2/69；设备进行NAT转换后的源IP和源端口为218.197.70.12/1048。

Initiator:
Source IP/Port : 192.168.200.2/2428
Dest IP/Port : 218.197.70.2/69
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port : 218.197.70.2/69
Dest IP/Port : 218.197.70.12/1048
VPN-Instance/VLAN ID/VLL ID:
Pro: UDP (17)　App: TFTP　State: UDP-OPEN
Start time: 2011-05-28 13:17:49　TTL: 119s
Root　Zone (in) : Private
Zone (out) : Public
Received packet (s) (Init) : 2 packet (s) 88 byte (s)
Received packet (s) (Reply) : 0 packet (s) 0 byte (s)

再让我们来看看PC2回应的ICMP差错报文的结构，如图8和图9所示：

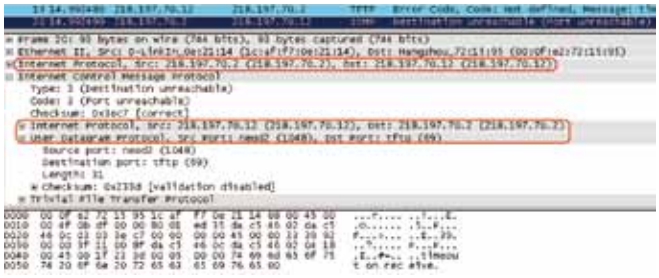


图8 外网ICMP差错报文结构

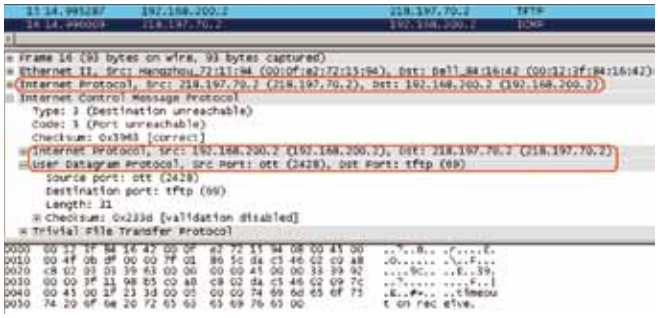


图9 内网ICMP差错报文结构

比较内网和外网的报文之后答案已经非常清楚了，设备根据ICMP差错报文体中的IP地址和端口号匹配设备上的NAT表，再根据这个NAT表项对报文目的IP，以及报文体中的源IP和源端口进行了转换，然后发送到内网的PC1。

同时设备会根据ICMP差错报文对相应的会话进行加速老化，已达到节省设备资源的目的。刚才老化时间还是120秒的表项在收到ICMP差错报文后迅速将老化时间调整为15秒。会话状态也由UDP-OPEN变为了Accelerate。

Initiator:
Source IP/Port : 192.168.200.2/2428
Dest IP/Port : 218.197.70.2/69
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port : 218.197.70.2/69
Dest IP/Port : 218.197.70.12/1048

```
VPN-Instance/VLAN ID/VLL ID:
Pro: UDP ( 17 )  App: TFTP  State: Accelerate
Start time: 2011-05-28 13:17:49  TTL: 14s
Root  Zone ( in ) : Private
      Zone ( out ) : Public
Received packet ( s ) ( Init ) : 9 packet ( s ) 403 byte ( s )
Received packet ( s ) ( Reply ) : 0 packet ( s ) 0 byte ( s )
```

分片报文的NAT转换

在PAT转换类型的NAT地址转换中，NAT除了对IP地址转换外，还使用到TCP或UDP报文的端口号、ICMP报文的ICMP头中的Identifier字段信息。当一个报文被分成若干片之后，这些信息只有首片报文会携带，后续分片报文依靠报文ID、分片标志位、分片偏移量依次关联到前一个分片。以ICMP报文为例，说明NAT对分片的IP报文进行的处理。在ICMP报文分片后，只有在首片ICMP报文中包含ICMP头的Identifier字段。在首片报文到达NAT设备后，按照正常的转换流程，根据源IP地址和Identifier信息生成转换表项并转发出去。在第二个及后续分片到达后，由于只包含IP地址却没有Identifier信息，可能因此无法进行NAT转换。解决的办法有两种：

■ 先重组，再进行NAT转换，在分片报文到达后，先进缓存，等属于这个IP报文的所有分片到达后进行虚拟分片重组，再进行NAT地址转换。最后将NAT转换完成的IP报文按照顺序发送出去。

■ 在首片到达并转换后，设备记录并保存转换首片使用的IP及Identifier信息，并在后续分片到达后应用同样的转换表项进行转换。

在H3C SecPath F1000-E中会由流分类和虚拟分片重组两个模块对分片报文进行处理。首先流分类会对到达设备的报文进行标识，如果是非首片的IP报文，流分类模块会将这个报文打上和其首片报文相同的标记，交给后续模块处理。NAT模块根据这个标识就可以判断如何对非首片的IP报文进行NAT转换了。

而虚拟分片重组所解决的就是报文乱序到达设备的情况，也就是使用上面的第一种方法对分片的报文进行NAT转换。

多核产品的无限连接

最后我们来介绍一下多核产品特有的NAT无限连接。

顾名思义，NAT无限连接就是内网通过NAT访问外网不会受到公网地址数量的影响，并发访问的数量只与设备的最大会话数相关。

前面我们讲到，设备在判断从外网进入内网的报文是否合法采用了五元组匹配，源地址、源端口号、协议类型、目的地址、目的端口号必须完全一样设备才会唯一确认一个表项。这里我们可以做一个实验，配置NAT设备的公网地址只有一个，并且配置成了高优先级。也就是说在内网所有的PC全部使用相同的协议访问外网同一台服务器、同一个端口的情况下，SecPath F1000-E可以支持64512个并发会话。

```
<SecPath F1000-E>display session statistics
Current session ( s ) :64514
Current  TCP session ( s ) : 1
Half-Open: 0  Half-Close: 0
Current  UDP session ( s ) : 64512
Current  ICMP session ( s ) : 0
Current  RAWIP session ( s ) : 0

Current relation table ( s ) : 0

session establishment rate:  0/s
TCP  session establishment rate:  0/s
UDP  session establishment rate:  0/s
ICMP  session establishment rate:  0/s
RAWIP  session establishment rate:0/s

Received  TCP:  100945 packet ( s )  22944126 byte ( s )
Received  UDP:  1222623 packet ( s )  63581139 byte ( s )
Received  ICMP:  630395 packet ( s )  17777039 byte ( s )
Received  RAWIP:  0 packet ( s )  0 byte ( s )
Dropped  TCP:  0 packet ( s )  0 byte ( s )
Dropped  UDP:  0 packet ( s )  0 byte ( s )
Dropped  ICMP:  0 packet ( s )  0 byte ( s )
Dropped  RAWIP:  0 packet ( s )  0 byte ( s )
```

如果访问的目的地址有两个，则并发会话数可以达到129024个：

```
<SecPath F1000-E>display session statistics
Current session ( s ) :129026
Current  TCP session ( s ) : 0
```

```
Half-Open: 0          Half-Close: 0
Current    UDP session ( s ) : 129024
Current    ICMP session ( s ) : 0
Current    RAWIP session ( s ) : 0

Current relation table ( s ) : 0

session establishment rate:       0/s
TCP    session establishment rate:       0/s
UDP    session establishment rate:       0/s
ICMP   session establishment rate:       0/s
RAWIP  session establishment rate:       0/s
```

Received	TCP:	150635 packet (s)	62257395 byte (s)
Received	UDP:	2940678 packet (s)	152925584 byte (s)
Received	ICMP:	630395 packet (s)	17777039 byte (s)
Received	RAWIP:	0 packet (s)	0 byte (s)
Dropped	TCP:	0 packet (s)	0 byte (s)
Dropped	UDP:	0 packet (s)	0 byte (s)
Dropped	ICMP:	0 packet (s)	0 byte (s)
Dropped	RAWIP:	0 packet (s)	0 byte (s)

从下面的详细会话表项可以看到，经过NAT转换后，相同的源地址＋端口可以与不同的目的地址＋端口建立会话对应关系。

```
Initiator:
Source IP/Port : 192.168.200.3/52024
Dest IP/Port   : 218.197.70.3/5000
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port : 218.197.70.3/5000
Dest IP/Port   : 218.197.70.200/65535
VPN-Instance/VLAN ID/VLL ID:
Pro: UDP ( 17 )   App: unknown        State: UDP-OPEN
Start time: 2011-05-28 17:24:12  TTL: 9881s
Root                Zone ( in ) : Private
                    Zone ( out ) : Public
Received packet ( s ) ( Init ) : 2 packet ( s ) 104 byte ( s )
Received packet ( s ) ( Reply ) : 0 packet ( s ) 0 byte ( s )

Initiator:
Source IP/Port : 192.168.200.3/50215
Dest IP/Port   : 218.197.70.4/5000
VPN-Instance/VLAN ID/VLL ID:
Responder:
Source IP/Port : 218.197.70.4/5000
Dest IP/Port   : 218.197.70.200/65535
VPN-Instance/VLAN ID/VLL ID:
Pro: UDP ( 17 )   App: unknown        State: UDP-OPEN
Start time: 2011-05-28 17:18:28  TTL: 9697s
Root                Zone ( in ) : Private
                    Zone ( out ) : Public
Received packet ( s ) ( Init ) : 6 packet ( s ) 312 byte ( s )
Received packet ( s ) ( Reply ) : 0 packet ( s ) 0 byte ( s )
```

而在真实的网络中，内网PC访问的目的IP和端口的组合可能有几千甚至上万个。那么NAT所能支持的并发数量也就变成了64512与目的地址数量的乘积，在理论上已经接近无限个了。

小结

虽然NAT可以解决IP地址空间不足，也可以很好地隐藏内部网络的拓扑结构，使网络更安全。但是它毕竟修改了报文的内容，随着应用场景的不断增多，相信还有很多需要NAT进行特殊处理的地方，这些地方有待于我们继续挖掘、探讨，使得NAT技术更加完善。

UPnP基本原理以及在NAT中的应用

文/王军

随着计算机产业以及计算机网络技术的迅猛发展，越来越多嵌入式设备的出现和家庭网络的发展，实现各种设备的互联互通已经成为人们的迫切需求，而实现家庭网络互联互通的关键是家庭网络的中间件技术。业界各大厂商都提出了自己的解决方案，其中以微软提出的UPnP最具有发展前途，也获得了最广泛的支持，目前UPnP基本是家庭网络设备必须支持的特性之一。

UPnP是通用即插即用（Universal Plug and Play）的缩写，主要用于设备的智能互联互通，使用UPnP协议不需要设备驱动程序，它可以运行在目前几乎所有的操作系统平台上，使得在办公室、家庭和其他公共场所方便地构建设备互联互通成为可能。

本文介绍了UPnP所定义的基本协议（如SSDP、GENA、SOAP等），重点分析了UPnP实现的基本工作流程，并通过抓包工具捕获数据包，对各种流程传递的协议报文进行详尽分析，最后结合NAT技术，重点叙述UPnP在NAT技术中的应用。

UPnP的结构规范

UPnP最大的愿景是希望任何设备一旦连接上网络，所有在网络上的设备马上就能知道有新设备加入，这些设备彼此之间能互相通信，更能直接使用或者控制它，一切都不需要人工设置，完全的即插即用。

UPnP的基本组件

服务、设备和控制点是UPnP网络的基本组件，它们之间的关系图如图1所示：

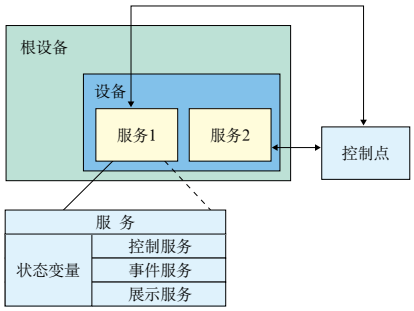


图1 UPnP组件图

设备（Device）

UPnP网络中定义的设备具有很广泛的含义，各种各样的家电、电脑外设、智能设备、无线设备、个人电脑等等都可以称之为设备。一台UPnP设备可以是多个服务的载体或多个子设备的嵌套。

服务（Service）

在UPnP网络中，最小的控制单元就是服务。服务描述的是指设备在不同情况下的动作和设备的状态。例如，时钟服务可以表述为时间变化值、当前的时间值以及设置时间和读取时间两个活动，通过这些动作，就可以控制服务。

控制点（Control Point）

在UPnP网络中，控制点指的是可以发现并控制其他设备的控制设备。在UPnP网络中，设备可以和控制点合并，为同一台设备，同时具有设备的功能和控制点的功能，即可以作为设备提供服务，也可以作为控制点发现和控制其他设备。

UPnP的部分术语

UUID

UUID含义是通用唯一识别码（Universally Unique Identifier），其目的是让分布式系统中的所有元素都有唯一的标识，其格式为xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx（8-4-4-16），分别表示当前的日期、时间、始终序列、全局唯一的IEEE机器标识，如果有网卡，则从网络的MAC地址获取，没有网卡则以其他方式获得。

UDN

单一设备名字（Unique Device Name），基于UUID，表示一个设备，在不同的时间，对于同一台设备此值应该是唯一的。

URI

Web上可用的每种资源，包括HTML文档、图像、视频片段、程序等，由一个通用资源标志符（Universal Resource Identifier，简称”URI”）进行定位。URI一般有三部分组成：访问资源的命名机制、存在资源的主机名、资源自身的名称，由路径表示。考虑下面的URI，它表示了当前的HTML 4.0规范：http://www.webmonkey.com.cn/html/html40/它表示一个可通过HTTP协议访问的资源，位于主机www.webmonkey.com.cn上，通过路径“/html/html40”访问。

URL

URL是URI命名机制的一个子集，URL是Uniform Resource Location的缩写，译为“统一资源定位符”。形象点说，URL是Internet上用来描述信息资源的字符串，主要用在各种www客户程序和服务器程序上，采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。

URN

URN是URL的一种更新形式，统一资源名称（Uniform Resource Name）。唯一标识一个实体的标识符，但是不能给出实体的位置。URN可以提供一种机制，用于查找和检索定义特定命名空间的架构文件。尽管普通的URL可以提供类似的功能，但是URN更强大更容易管理，因为它可以引用多个URL。

UPnP协议栈

UPnP定义了设备之间、设备和控制点、控制点之间通信的协议。完整的UPnP有设备寻址、设备发现、设备描述、设备控制、事件通知和基于HTML的描述等几部分构成。UPnP设备协议栈如图2所示：

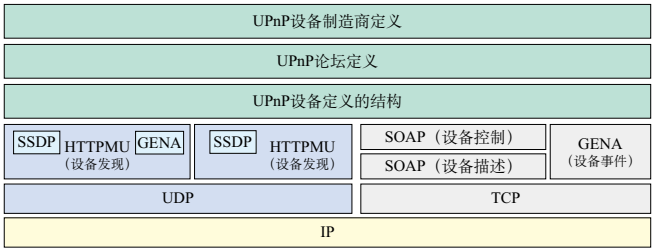


图2 UPnP协议栈

UPnP协议结构最底层的TCP/IP协议是UPnP协议结构的基础。IP层用于数据的发送与接收。对于需要可靠传送的信息，使用TCP进行传送，反之则使用UDP。UPnP对网络的底层没有要求，可以是以太网、WiFi、IEEE1394等等，只需支持IP协议即可。

构建在TCP/IP协议之上的是HTTP协议及其变种，这一部分是UPnP的核心，所有UPnP消息都被封装在HTTP协议及其变种中。HTTP协议的变种是HTTPU和HTTPMU，这些协议的格式沿袭了HTTP协议，只不过与HTTP不同的是他们通过UDP而非TCP来承载的，并且可用于组播进行通信。

SSDP协议

简单服务发现协议（Simple Service Discovery Protocol：SSDP），是内建在HTTPU/HTTPMU里，定义如何让网络上有的服务被发现的协议。具体包括控制点如何发现网络上有哪些服务，以及这些服务的资讯，还有控制点本身宣告他提供哪些服务。该协议运用在UPnP工作流程的设备发现部分。

SOAP协议

简单对象访问协议（Simple Object Access Protocol：SOAP）定义如何使用XML与HTTP来执行远程过程调用（Remote Procedure Call）。包括控制点如何发送命令消息给设备，设备收到命令消息后如何发送响应消息给控制点。该协议运用在UPnP工作流程的设备控制部分。

GENA协议

通用事件通知架构（Generic Event Notification Architecture：GENA）定义在控制点想要监听设备的某个服务状态变量的状况时，控制点如何传送订阅信息并如何接收这些信息，该协议运用在UPnP工作流程的事件订阅部分。

UPnP实现的工作流程

图3是UPnP的运行流程，我们先大概介绍下：

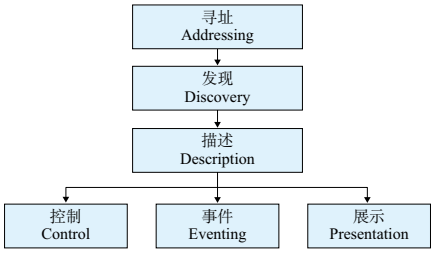


图3 UPnP的运行流程

- 首先控制点和设备都先获取IP地址后才能进行下一步的工作；
- 控制点首先要寻找整个网络上的UPnP设备，同时网络上的设备也要宣告自身的存在；
- 控制点要取得设备的描述，包括这些设备提供什么样的服务；
- 控制点发出动作信息给设备；
- 控制点监听设备的状态，当状态改变时作出相应的处理动作。

寻址（Addressing）

UPnP网络的基础是TCP/IP，这就决定了每一个UPnP组件必须有IP地址。一台UPnP设备寻址的一般过程是：首先向DHCP服务器发送DHCP Discover的消息，如果在指定的时间内，设备没有收到DHCP Offer回应消息，设备必须使用AUTO-IP完成IP地址的获取。当然也可以使用静态配置的IP地址。

发现（Discovery）

连接到网络上的设备确定了IP地址之后，就会进入发现操作阶段。设备发现是UPnP实现的第一步。设备发现是由简单发现协议SSDP来完成的。当一台设备加入到网络中，发现过程允许设备向网络上的控制节点告知它提供的服务，当一个控制点加入到网络中，设备发现过程允许控制点寻找网络上感兴趣的设备。在这两种情况下，基本的交换信息就是发现消息。发现消息包括设备的一些特定信息或者某项服务的信息，例如它的类型、标志符、等等。图4是发现流程的框架图：

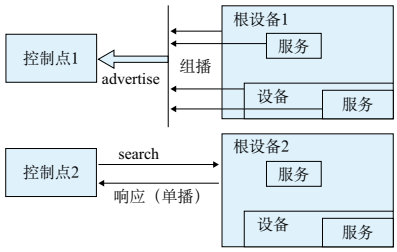


图4 发现过程框架图

描述（Description）

UPnP的第二步是设备描述。在控制点发现一台设备后，控制点对该设备可能仅仅知道设备或者服务的UPnP类型，设备的UUID和设备描述的URL地址，还需要知道更多的信息。控制点可以从发现消息中得到设备描述的URL，通过URL取回设备描述的信息。设备描述的一般过程图如图5所示：

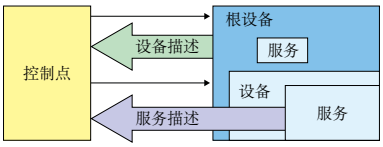


图5 设备描述以及服务描述

设备描述

UPnP对某一设备的描述以XML形式来表示，设备描述包括制造商信息、模块名称和编号、序列号等等。对于一个物理设备可以包含多个逻辑设备，多个逻辑设备既可以是一个根设备其中嵌入多个设备，也可以是多个根设备的方式存在。设备描述由设备制造商提供，采用XML描述，遵循UPnP框架协议。

服务描述

服务的描述包含一系列内容，具体有服务运行时刻的状态，运行时间等等。服务描述也由设备制造商提供，采用XML描述，遵循UPnP框架协议。

控制（Control）

在接收设备和服务描述之后，控制点可以向这些服务发出动作，同时控制点也可以轮询服务的当前状态。控制点将动作发送到设备服务，在动作完成或者失败后，服务返回相应的结果或者错误信息。其基本过程如图6所示：

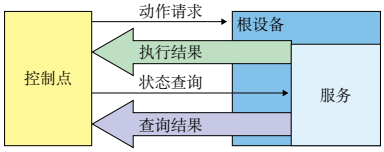


图6 控制过程示意图

为了控制一台设备，控制点向设备服务发出一动作，这一般是由控制点向服务的控制URL地址发送一个适当的控制消息。而服务则会对此动作做出响应，返回相关的结果或错误。

事件（Even ting）

如上文的描述部分所述，一个即插即用服务描述包括服务响应的动作列表和运行时描述服务状态的变量列表。如果一个或多个状态被事件触发，服务将会在这些状态发生变化时发布更新，控制点可以订阅以获得此信息。在事件机制中，发布者指事件的来源（通常为设备服务），订阅者指事件目的地（通常为控制点）。

要订阅事件，订阅者可发送一条请求订阅消息。它将以这个订阅到持续时间作为响应。要保持订阅，订阅者必须在订阅过期之前进行续订。当订阅者不再需要发布者发送的事件时，订阅者应当取消其订阅。

发布者通过发送事件消息提醒订阅者状态改变。在订阅者第一次订阅时，需要发送一个专门的初始化事件消息。该事件消息包含所有事件的名称和值，并且允许订阅者初始化其服务状态。为了支持多个控制点，在动作生效后所有订阅者均会收到通知。由此，将向所有订阅者发送全部事件消息。事件消息使用HTTP协议传送，事件详细定义在通用事件通知结构（GENA）协议中。

展示（Presentation）

在控制点发现设备和取得设备描述之后，展示也就开始了。如果设备拥有进行展示的URL，那么控制点就可以通过此URL取得一个页面，在浏览器中加载该页面，并根据页面功能，支持用户控制设备或浏览设备状态。每一项完成的程度取决于展示页面和设备的具体功能。

设备展示包含在设备描述的Presentation URL字段。设备展示可以完全由设备制造商提供，它采用HTML页的形式，使用HTTP进行发布。图7是展示的流程示意图：

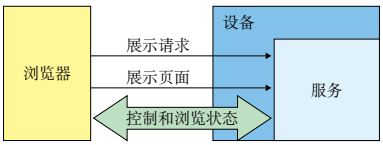


图7 展示示意图

UPnP在NAT中的应用

应用场景

如果用户是通过NAT接入Internet的，同时需要使用BC、电骡eMule等P2P这样的软件，这时UPnP功能就会带来很大的便利。利用UPnP能自动的把BC、电骡eMule等侦听的端口号映射到公网上，以便公网上的用户也能对NAT私网侧发起连接。

实现UPnP所需条件

必须同时满足三个条件：

■ NAT网关设备必须支持UPnP功能；

■ 操作系统必须支持UPnP功能；我们常见的Windows XP是支持UPnP的；

■ 应用软件必须支持UPnP功能；比如BC、电骡eMule、MSN软件都是支持的。

以上三个条件必须同时满足，缺一不可。

NAT网关设备的设置

NAT网关设备的UPnP功能，不同的型号设置界面略有不同，有的设备是默认使能UPnP功能的，有的设备是需要手工使能，不同的设备界面和方法可能略有不同，某款设备的设置具体如下：

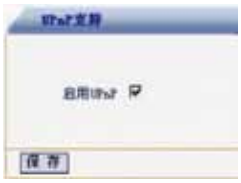


图8 网关设备上的UPnP使能

操作系统的UPnP功能设置

重点描述下Windows XP的UPnP功能如何启用。如果使用的是Windows XP SP2版本的系统，首先进入：控制面板——添加或删除程序——添加/删除Windows组件中，在“网络服务”中勾选“UPnP用户界面”，如图9所示：

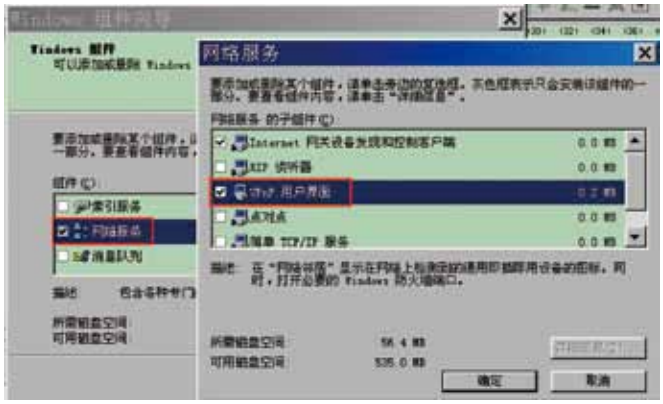


图9 UPnP安装截图

确定后，系统会自动安装相应的组件，可能会提示你插入安装光盘，按照提示完成即可。接着打开Windows自带的防火墙，在“例外”选项中勾选“UPnP框架”，如图10所示：

图10 UPnP中防火墙的设置

接下来，在Windows中打开相应的UPnP服务：

进入“控制面板——管理工具——服务”，找到SSDP Discovery Service和Universal Plug and Play Device Host两项服务，选择启动即可。设置完以上后，就可以在“网络连接”中看到多了Internet网关，这表明添加UPnP已经成功了，如图11所示：

图11 UPnP成功后的网络接口

在应用层的设置就简单了，以BitComet v1.02版本为例来说明下。在选项——高级设置中，把“允许使用UPnP自动端口映射”前勾上。如图12所示：

图12 BT软件中的端口映射

在“全局统计”页签中可以看到NAT端口映射已添加。这样在进行P2P下载时，在公网侧的其他用户也可以向处于私网中的用户发起连接，可以大大的提高下载效率与速度。如图13所示：

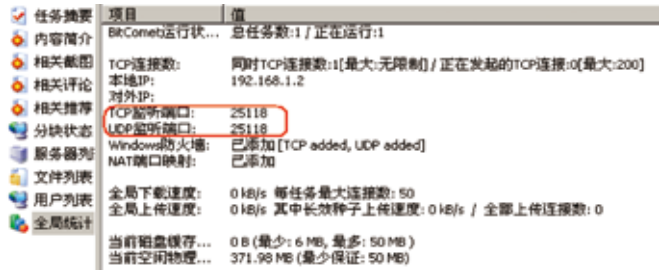


图13 BT软件中侦听的端口号

点击“Internet连接”右键属性，再点设置，可以看到BitComet所侦听的TCP和UDP端口已经被映射出来了。如图14所示：

图14 成功映射的端口号

接下来我们来抓包分析，NAT的端口映射如何通过UPnP来自动完成的?

发现阶段的报文交互

设备发现是UPnP网络实现的第一步，设备发现是由简单发现协议SSDP（Simple Service Discovery Protocol）来定义的。设备发现分两种情况：

主动告知

设备加入到网络中，设备发现过程允许设备向网络上的控制点告知它提供的服务，并且定期发送。

NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: search target（服务类型）
NTS: ssdp:alive
USN: advertisement UUID（不同服务的统一服务名，标示相同类型服务能力）

通过Ethereal抓包可以发现，支持UPnP的网关设备启动后会发出一系列SSDP的alive消息，用于向外通告自己。图15是设备发出的一系列SSDP消息，图16是其中一条消息的详情。

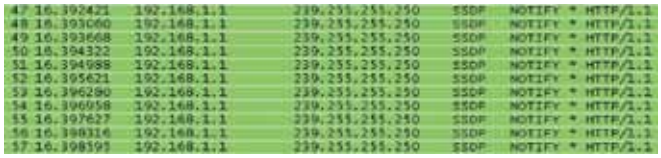


图15 SSDP消息截图一

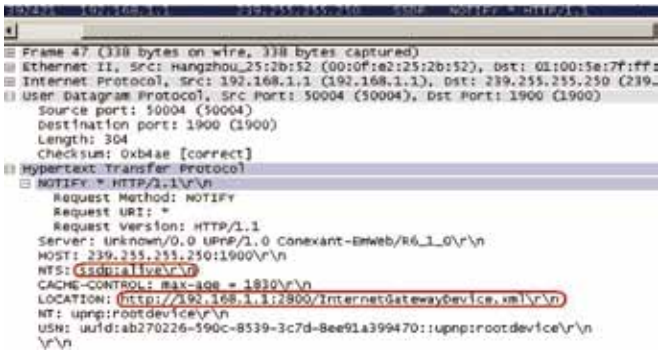


图16 SSDP消息截图二

注意：上图location字段是网关向外通告了自己的IP地址以及UPnP监听的端口号，后续由设备向端口号2800发起TCP连接，利用XML来进行后续的描述、表示、控制等操作。

利用查询来发现

控制点加入到网络中时，设备发现过程允许控制点寻找网络上感兴趣的设备，并使得控制点获得设备能力的描述，同时控制点也可以向设备发送命令，侦听设备状态的改变，并将设备展示给用户。

查询消息

M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: ssdp:discover
Mx:seconds to delay response
ST:search target

各个参数所表达的含义：

Mx:	设置设备响应最长等待时间，设备响应在0和这个值之间随机选择响应延迟的值
ST:	设置服务查询的目标，它必须是下面的类型：
	ssdp:all 搜索所有设备和服务
	upnp:rootdevice 仅搜索网络中的根设备
	uuid:device-UUID 查询UUID标识的设备
	urn:schemas-upnp-org:device:device-
	type:version查询device-Type字段指定的设备类型，设备类型和版本由UPnP组织定义
	urn:schemas-upnp-org:service:service-
	type:version 查询service-Type字段指定的服务类型，服务类型和版本由UPnP组织定义

图17是PC向外发送的查询报文：

图17 PC向外发送的查询报文截图

响应消息

HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when reponse was generated
EXT:
LOCATION: URL for UPnP description for root device
Server: OS/Version UPnP/1.0 product/version
ST: search target
USN: advertisement UUID

各参数的意义：

max-age	指定通知消息存活时间，如果超过此时间间隔，控制点可以认为设备不存在
DATE	指定响应生成的时间
EXT	向控制点确认MAN头域已经被设备理解
LOCATION	包含根设备描述得URL地址
Server	包含操作系统名，版本，产品名和产品版本信息
ST	内容和意义与查询请求的相应字段相同
USN	表示不同服务的统一服务名，它提供了一种标识出相同类型服务的能力

通过Ethereal抓包得到的响应消息，如图18所示：

图18 响应消息截图

当在网关设备上去使能UPnP功能时，设备会向外发送SSDP的byebye消息。

SSDP：byebye消息
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
NT:search target
NTS: ssdp:byebye
USN: advertisement UUID

发出的SSDP的byebye利用Ethereal抓包解析，具体如图19所示：

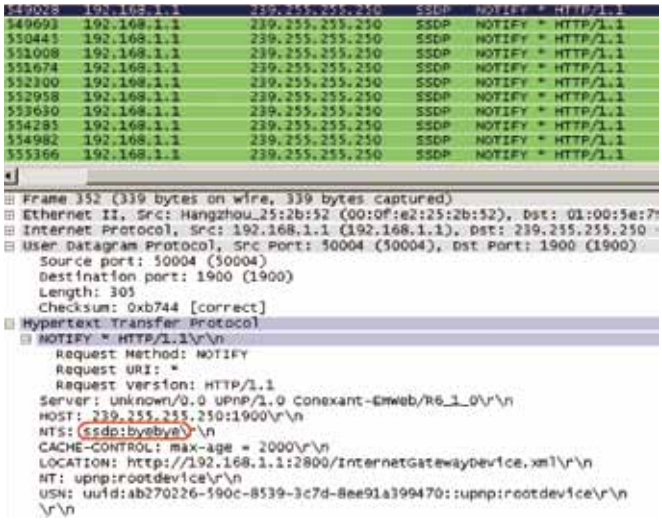


图19 SSDP的byebye消息截图

描述阶段的报文交互

UPnP网络结构的第二步是设备描述。在控制点发现了一个设备之后，控制点仍然对设备知之甚少，控制点可能仅仅知道设备或服务的UPnP类型，设备的UUID和设备描述的URL地址。为了让控制点更多的了解设备和它的功能或者与设备交互，控制点必须从发现消息中得到设备描述的URL，通过URL取回设备描述。

设备描述是由设备制造商提供的，采用XML表述，并且遵循UPnP设备模版。此模版是由UPnP工作委员会生成的。设备描述包括制造商信息，包括模块名称和编号，序列号，制造商名称，制造商网站的URL等。对于一个物理设备可以包含多个逻辑设备，多个逻辑设备既可以是一个根设备其中嵌入多个设备，也可以是多个根设备的方式实现。

其中一个描述所建立的TCP连接如图20所示：

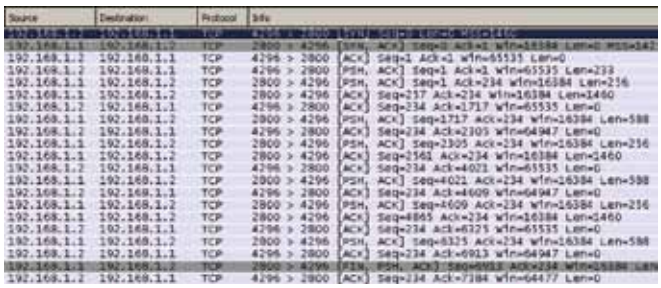


图20 建立的TCP连接截图

用Ethereal的Follow TCP Stream来解析上面的报文：

```
GET /InternetGatewayDevice.xml HTTP/1.1
Accept: text/xml，application/xml
User-Agent: Mozilla/4.0（compatible；UPnP/1.0；Windows NT/5.1）
Host: 192.168.1.1:2800
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

HTTP/1.1 200 OK
Server:Unknown/0.0 UPnP/1.0 Conexant-EmWeb/R6_1_0
Connection: close Content-Type: text/xml
Expires: Thu，01 Jan 1970 00:00:00 GMT
Cache-Control: no-cache Pragma: no-cache

<?xml version = "1.0"?>
<root xmlns = "urn:schemas-upnp-org:device-1-0">
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
<URLBase>http://192.168.1.1:2800/</URLBase>
<device>
<deviceType>urn:schemas-upnp-org:device:InternetGatewayDevice:1</deviceType>
<friendlyName>Huawei-3Com IGD</friendlyName>
<manufacturer>Huawei-3Com</manufacturer>
<manufacturerURL>http://aolynk.huawei-3com.com/</manufacturerURL>
<modelDescription>Huawei-3Com UPnP IGD in ISOS 9.0.8.9</modelDescription>
<modelName>IGD</modelName>
<modelNumber>9.0.8.9</modelNumber>
<modelURL>http://www.vendor.com/model</modelURL>
<serialNumber>Prototype</serialNumber>
<UDN>uuid:ab270226-590c-8539-3c7d-8ee91a399470</UDN>
<UPC>Universal Product Code</UPC>
<iconList>
<icon>
……
```

控制阶段的报文交互

设备控制是UPnP网络的第三步。控制点先发送一个控制行为请求，要求设备开始服务，然后再按设备的URL发送相应的控制消息，控制消息是放置在XML文件中的那些SOAP格式的信息，最后，服务返回响应信息，指出服务是成功或是失败。对其中一个控制阶段所建立的TCP连接的如图21所示：

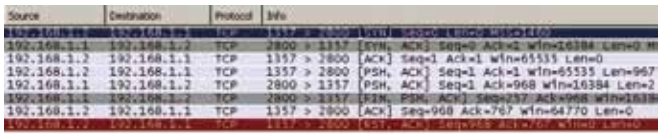


图21 控制阶段所建立的TCP连接截图

用Ethereal的Follow TCP Stream来解析上面的报文：

```
POST /EmWeb/UPnP/Control/4 HTTP/1.1
Content-Type: text/xml；charset = "utf-8"
SOAPAction: "urn:schemas-upnp-org:service:WANIPConnection:1#GetSpecificPortMapping
Entry"
User-Agent: Mozilla/4.0（compatible；UPnP/1.0；Windows 9x）
Host: 192.168.1.1:2800
Content-Length: 625
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache
<?xml version = "1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:
Body><m:GetSpecificPortMappingEntryxmlns:m = "urn:schemas-upnp-org:service:WANIP
Connection:1"><NewRemoteHostxmlns:dt = "urn:schemas-microsoft-com:datatypes"dt:dt
= "string"></NewRemoteHost><NewExternalPortxmlns:dt = "urn:schemas-microsoft-
com:datatypes"dt:dt = "ui2">25118</NewExternalPort><NewProtocolxmlns:dt =
"urn:schemas-microsoft-com:datatypes"dt:dt = "string">TCP</NewProtocol></
m:GetSpecificPortMappingEntry></SOAP-ENV:Body></SOAP-ENV:Envelope>

HTTP/1.1 200 OK
Server: Unknown/0.0 UPnP/1.0 Conexant-EmWeb/R6_1_0
Connection: close
Content-Type: text/xml；charset = utf-8
Expires: Thu，01 Jan 1970 00:00:00 GMT
Cache-Control: no-cache
Pragma: no-cache

<s:Envelope
xmlns:s = "http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
```

```
<u:GetSpecificPortMappingEntryResponse xmlns:u = "urn:schemas-upnp-org:service:WANIPConnection:1"><NewInternalPort>25118</NewInternalPort><NewInternalClient>192.168.1.2</NewInternalClient><NewEnabled>1</NewEnabled><NewPortMappingDescription>BitComet （ 192.168.1.2:25118 ） 25118 TCP</NewPortMappingDescription><NewLeaseDuration>0</NewLeaseDuration></u:GetSpecificPortMappingEntryResponse>
</s:Body>
</s:Envelope>
```

通过上面可以看出，PC利用通过UPnP向网关发送的SOAP格式的控制消息，请求BitComet的TCP的25118端口映射，网关返回OK消息。

事件阶段和展示阶段的报文交互

设备事件是UPnP网络的第四步。一个服务的UPnP描述包括服务响应的动作列表和运行时模拟服务状态的变量列表。当这些变量改变时，就会产生一个事件，系统将修改事件列表的内容，事件服务器将事件向整个网络广播。控制点也可以事先向事件服务器预约事件信息，保证控制点感兴趣的事件及时准确的传送过来。事件消息放在XML文件中，格式是GENA。在UPnP的自动端口映射没有该报文的交互。

```
NOTIFY delivery path HTTP/1.1
HOST: delivery host:delivery port
CONTENT-TYPE: text/xml
NT: upnp:event
NTS: upnp:propchange
SID: uuid:subscription-UUID
SEQ: event key
<e:propertyset xmlns:e = "urn:schemas-upnp-org:event-1-0">
<e:property>
<variableName>new value</variableName>
</e:property>
Other variable names and values （ if any ） go here
</e:propertyset>
```

在展示阶段，UPnP的自动端口映射也没有该报文的交互。在此就略去。

STUN/TURN技术浅析

文/顾雷雷

在现实Internet网络环境中，大多数计算机主机都位于防火墙或NAT之后，只有少部分主机能够直接接入Internet。很多时候，我们希望网络中的两台主机能够直接进行通信，即所谓的P2P通信，而不需要其他公共服务器的中转。由于主机可能位于防火墙或NAT之后，在进行P2P通信之前，我们需要进行检测以确认它们之间能否进行P2P通信以及如何通信。这种技术通常称为NAT穿透（NAT Traversal）。最常见的NAT穿透是基于UDP的技术，如RFC3489中定义的STUN协议。

STUN，首先在RFC3489中定义，作为一个完整的NAT穿透解决方案，英文全称是Simple Traversal of UDP Through NATs，即简单的用UDP穿透NAT。

在新的RFC5389修订中把STUN协议定位于为穿透NAT提供工具，而不是一个完整的解决方案，英文全称是session Traversal Utilities for NAT，即NAT会话穿透效用。RFC5389与RFC3489除了名称变化外，最大的区别是支持TCP穿透。

TURN，首先在RFC5766中定义，英文全称是Traversal Using Relays around NAT：Relay Extensions to session Traversal Utilities for NAT，即使用中继穿透NAT：STUN的扩展。简单的说，TURN与STURN的共同点都是通过修改应用层中的私网地址达到NAT穿透的效果，异同点是TURN是通过两方通讯的“中间人”方式实现穿透。

STUN

了解STUN之前，我们需要了解NAT的种类。

NAT对待UDP的实现方式有4种，分别如下：

Full Cone NAT

完全锥形NAT，所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号，并且任何一个外网主机都可以通过这个映射的外网IP和端口号向这台内网主机发送包。

Restricted Cone NAT

限制锥形NAT，它也是所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号。与完全锥形不同的是，外网主机只能够向先前已经向它发送过数据包的内网主机发送包。

Port Restricted Cone NAT

端口限制锥形NAT，与限制锥形NAT很相似，只不过它包括端口号。也就是说，一台IP地址X和端口P的外网主机想给内网主机发送包，必须是这台内网主机先前已经给这个IP地址X和端口P发送过数据包。

Symmetric NAT

对称NAT，所有从同一个内网IP和端口号发送到一个特定的目的IP和端口号的请求，都会被映射到同一个IP和端口号。如果同一台主机使用相同的源地址和端口号发送包，但是发往不同的目的地，NAT将会使用不同的映射。此外，只有收到数据的外网主机才可以反过来向内网主机发送包。

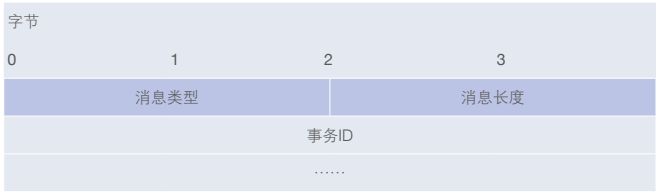
RFC3489/STUN

STUN（Simple Traversal of User Datagram Protocol Through Network Address Translators），即简单的用UDP穿透NAT，是个轻量级的协议，是基于UDP的完整的穿透NAT的解决方案。它允许应用程序发现它们与公共互联网之间存在的NAT和防火墙及其他类型。它也可以让应用程序确定NAT分配给它们的公网IP地址和端口号。STUN是一种Client/Server的协议，也是一种Request/Response的协议，默认端口号是3478。

报文结构

消息头

所有的STUN消息都包含20个字节的消息头，包括16位的消息类型，16位的消息长度和128位的事务ID。



消息类型许可的值如下：

0×0001：捆绑请求

0×0101：捆绑响应

0×0111：捆绑错误响应

0×0002：共享私密请求

0×0102：共享私密响应

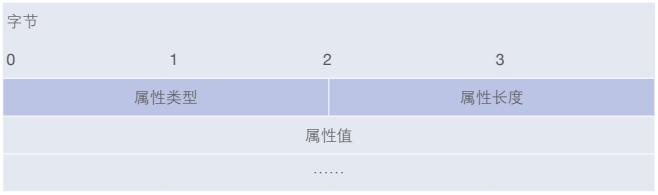
0×0112：共享私密错误响应

消息长度，是消息大小的字节数，但不包括20字节的头部。

事务ID，128位的标识符，用于随机请求和响应，请求与其相应的所有响应具有相同的标识符。

消息属性

消息头之后是0或多个属性，每个属性进行TLV编码，包括16位的属性类型、16位的属性长度和变长属性值。



属性类型定义如下：

■ 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位的端口号及长度固定的IP地址。

■ RESPONSE-ADDRESS

RESPONSE-ADDRESS属性表示响应的目的地址。

■ CHASNGE-REQUEST

客户使用32位的CHANGE-REQUEST属性来请求服务器使用不同的地址或端口号来发送响应。

■ 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。

■ CHANGED-ADDRESS

如果捆绑请求的CHANGE-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。

■ USERNAME

username属性用于消息的完整性检查，用于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。当使用消息完整性检查时，可有选择地出现在捆绑请求中。

■ PASSWORD

password属性用在共享私密响应中，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字节的倍数，以保证属性与边界对齐。

■ 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；MESSAGE-INTEGRITY属性必须是任何STUN消息的最后一个属性。它的内容决定了HMAC输入的Key值。

■ 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。它的响应号数值范围从100到699。

下面的响应号，与它们缺省的原因语句一起，目前定义如下：

400（错误请求）：请求变形了。客户在修改先前的尝试前不应该重试该请求。

401（未授权）：捆绑请求没有包含MESSAGE-INTERITY属性。

420（未知属性）：服务器不认识请求中的强制属性。

430（过期资格）：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（完整性检查失败）：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（丢失用户名）：捆绑请求包含MESSAGE-INTEGRITY属性，但没有username属性。完整性检查中两项都必须存在。

433（使用TLS）：共享私密请求已经通过TLS（Transport Layer Security，即安全传输层协议）发送，但没有在TLS上收到。

500（服务器错误）：服务器遇到临时错误，客户应该再次尝试。

600（全局失败）：服务器拒绝完成请求，客户不应该重试。

■ 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。

■ 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。

属性空间分为可选部分与强制部分，值超过0×7fff的属性是可选的，即客户或服务器即使不认识该属性也能够处理该消息；值小于或等于0×7fff的属性是强制理解的，即除非理解该属性，否则客户或服务器就不能处理该消息。

实现原理

图1 STUN

STUN协议的完整交互过程如上，下面我们来介绍具体实现步骤。

一般情况下，客户会配置STUN服务器提供者的域名，该域名被解析为IP地址和SRV过程的端口号。服务器名是“STUN”，使用UDP协议发送捆绑请求，使用TCP协议发送共享私密请求。STUN协议的缺省端口号为3478。

若要提供完整性检查，STUN在客户和服务器间使用128位的共享私密，作为在捆绑请求和捆绑响应中的密匙。

首先，客户通过发现过程获得它将与之建立TCP连接的IP地址和端口号。客户打开该地址和端口的连接，开始TLS协商，验证服务器的标识。客户发送共享私密请求。该请求没有属性，只有头。服务器生成响应。

客户会在该连接上生成多个请求，但在获得用户名和密码后关闭该连接。

服务器收到共享私密请求，验证从TLS连接上到达的该请求；如果不是通过TLS收到的请求，则生成共享私密错误响应，并设置ERROR-CODE属性为响应号433；这里区分两种情况：若通过TCP收到请求，则错误响应通过收到请求的相同连接发送；若通过UDP收到请求，则错误响应发送回请求送出的源IP和端口。

服务器检查请求中的任何属性，当其中有不理解的小于或等于0×7fff的值，则生成共享私密错误响应，设置ERROR-CODE属性为响应号420，并包括UNKNOWN-ATTRIBUTE属性，列出它不理解的小于或等于0×7fff的属性的值。该错误响应通过TLS连接发送。

若请求正确，服务器创建共享私密响应，包含与请求中相同的事务ID，并包含username和password属性。用户名在10分钟内有效。

共享私密响应通过与收到请求的相同的TLS连接发送，服务器保持连接打开状态，由客户关闭它。

接着，客户发送捆绑请求，携带的属性包括：

■ 可选属性：RESPONSE-ADDRESS属性和CHANGE-REQUEST属性；

■ 强制属性：MESSAGE-INTEGRITY属性和username属性。

客户发送捆绑请求，通过客户重传来提供可靠性。客户开始用100ms的间隔重传，每次重传间隔加倍，直至1.6秒。之间间隔1.6秒的重传继续，直到收到响应或总共已经发送了9次。因此，若9500ms后，还未收到响应，客户认为传输已经失败。

服务器检查捆绑请求的MESSAGE-INTEGRITY属性，不存在则生成捆绑错误响应，设置ERROR-CODE属性为响应号401；若存在，计算请求的HMACKey值。

服务器检查username属性，不存在则生成捆绑错误响应，设置ERROR-CODE属性为响应号432；若存在，但不认识该username的共享私密（例如，它超时了），生成捆绑错误响应，设置ERROR-CODE属性为响应号430。

若服务器知道该共享私密，但所计算的HMAC与请求的不同，生成捆绑错误响应，设置ERROR-CODE属性为响应号431。

假设消息完整性检查通过了，服务器检查请求中的任何属性的值，若遇到不理解的小于或等于0×7fff的值，生成捆绑错误响应，设置ERROR-CODE属性为响应号420，该响应包含UNKNOWN-ATTRIBUTE属性，并列出不理解的小于或等于0×7fff的属性。

若请求正确，服务器生成单个捆绑响应，包含与捆绑请求相同的事务ID。服务器在捆绑响应中加入MAPPED-ADDRESS属性，该属性的IP地址和端口号为捆绑请求的源IP地址和端口号。

捆绑响应的源地址和端口号取决于捆绑请求中CHANGE-REQUEST属性的值及捆绑请求收到的地址和端口号相关。总结如下：

标志	源地址	源端口号	CHANGED-ADDRESS
无	Da	Dp	Ca:Cp
改变IP	Ca	Dp	Ca:Cp
改变端口号	Da	Cp	Ca:Cp
改变IP且改变端口号	Ca	Cp	Ca:Cp

表1 标志对数据包源和CHANGED-ADDRESS的影响

服务器在捆绑响应中加入SOURCE-ADDRESS属性，包含用于发送捆绑响应的源地址和端口号；加入CHANGED-ADDRESS属性，包含源IP地址和端口号。

如果捆绑请求中包含了username和MESSAGE-INTEGRITY属性，则服务器在捆绑响应中加入MESSAGE-INTEGRITY属性。

如果捆绑请求包含RESPONSE-ADDRESS属性，则服务器在捆绑响应中加入REFLECTED-FROM属性：如果捆绑请求使用从共享私密请求获得的用户名进行认证，则REFLECTED-FROM属性包含共享私密请求到达的源IP地址和端口号；若请求中的用户名不是使用共享私密分配的，则REFLECTED-FROM属性包含获得该用户名的实体的源IP地址和端口号；若请求中没有用户名，且服务器愿意处理该请求，则REFLECTED-FROM属性包含请求发出的源IP地址和端口号。

服务器不会重传响应，可靠性通过客户周期性地重发请求来保障，每个请求都会触发服务器进行响应。

客户端判断响应的类型是捆绑错误响应还是捆绑响应。捆绑错误响应通常在请求发送的源地址和端口收到；捆绑响应通常在请求中的RESPONSE-ADDRESS属性的地址和端口收到，若没有该属性，则捆绑响应将在请求发送的源地址和端口号收到。

■ 若是捆绑错误响应，客户检查响应中的ERROR-CODE属性的响应号：400至499之间的未知属性按属性400处理，500至599之间的未知属性按500处理，600至699之间的未知属性按600处理。任何100和399之间的响应都会使请求重传中止，但其他则忽略；若客户收到响应的属性类型大于0×7fff，则忽略该属性，若小于或等于0×7fff，则请求重传停止，并忽略整个响应；

■ 若是捆绑响应，客户检查响应的MESSAGE-INTEGRITY属性：如果不存在，客户在请求中加入MESSAGE-INTEGRITY属性，并放弃该响应；如果存在，客户计算响应的HMAC。如果计算出的HMAC与响应中的不同，则放弃该响应，并警告客户可能受到了攻击；若计算出的HMAC与响应中的匹配，则过程继续；

■ 不论收到捆绑响应还是捆绑错误响应，都将中止该请求的重传。客户在第一次响应后继续监听捆绑请求的响应10秒钟，如果这期间它收到任何消息类型不同的响应或不同的MAPPED-ADDRESS属性，它将警告用户可能受到攻击；并且，如果客户收到的捆绑响应次数超过它发送的捆绑请求数的两倍，它将警告用户可能受到攻击；若捆绑响应经过认证，上述攻击并未导致客户丢弃MAPPED-ADDRESS，则客户可以使用该MAPPED-ADDRESS和SOURCE-ADDRESS属性。

STUN功能举例

客户通过带外方式获得STUN服务器信息后，就打开对应的地址和端口的连接，并开始与STUN服务器进行TLS协商。一旦打开了连接，客户就通过TCP协议发送共享私密请求，服务器生成共享私

密响应。STUN在客户和服务器间使用共享私密，用作捆绑请求和捆绑响应中的密匙。之后，客户使用UDP协议向STUN服务器发送捆绑请求，当捆绑请求消息到达服务器的时候，它可能经过了一个或者多个NAT。结果是STUN服务器收到的捆绑请求消息的源IP地址被映射成最靠近STUN服务器的NAT的IP地址，STUN服务器把这个源IP地址和端口号复制到一个捆绑响应消息中，发送回拥有这个IP地址和端口号的客户端。

当STUN客户端收到捆绑响应消息之后，它会将自己发送捆绑请求时绑定的本地IP地址和端口号同捆绑响应消息中的IP地址和端口号进行比较，如果不匹配，就表示客户端正处于一个或者多个NAT的前面。

在Full-Cone NAT的情况下，在捆绑响应消息中的IP地址和端口是属于公网的，公网上的任何主机都可以使用这个IP地址和端口号向这个应用程序发送数据包，应用程序只需要在刚才发送捆绑请求的IP地址和端口上监听即可。

当然，客户可能并不在一个Full-Cone NAT的前面，实际上，它并不知道自己在一个什么类型的NAT的前面。为了确定NAT的类型，客户端使用附加的捆绑请求。具体过程是很灵活的，但一般都会像下面这样工作：客户端再发送一个捆绑请求，这次发往另一个IP地址，但是使用的是跟上一次同一个源IP地址和源端口号，如果返回的数据包里面的IP地址、端口号和第一次返回的数据包中的不同，客户端就会知道它是在一个对称NAT的前面。客户端为了确认自己是否在一个完全锥形NAT的前面，客户端可以发送一个带有标志的捆绑请求，这个标志告诉服务器使用另一个IP地址和端口发送捆绑响应。换句话说，如果客户端使X/Y的IP地址端口对向A/B的IP地址端口对发送捆绑请求，服务器就会使用源IP地址和源端口号为C/D的地址端口对向X/Y发送捆绑响应。如果客户端收到了这个响应，它就知道它是在一个Full-Cone NAT前面。

STUN协议允许客户端请求服务器从收到捆绑请求的IP地址往回发捆绑响应，但是要使用不同的端口号。这可以用来检查客户端是否在Port Restricted Cone NAT的前面还是在Restricted Cone NAT的前面。

RFC5389/STUN

STUN协议在RFC5389中被重新命名为session Traversal Utilities for NAT，即NAT会话穿透效用。在这里，NAT会话穿透效用被定位为一个用于其他解决NAT穿透问题协议的协议。它可以用于终端设备检查由NAT分配给终端的IP地址和端口号。同时，它也被用来检查两个终端之间的连接性，好比是一种维持NAT绑定表项的保活协议。STUN可以用于多种NAT类型，并不需要它们提供特殊的行为。

STUN本身不再是一种完整的NAT穿透解决方案，它相当于是一种NAT穿透解决方案中的工具。这是与RFC3489/STUN版本相比最重要的改变。

STUN用途

目前定义了三种STUN用途：

- Interactive Connectivity Establishment（ICE）【MMUSIC-ICE】，交互式连接建立；
- Client-initiated Connections for SIP【SIP-OUTBOUND】，用于SIP的客户端初始化连接；
- NAT Behavior Discovery【BEHAVE-NAT】，NAT行为发现。

报文结构

消息头

STUN消息头为20字节，后面紧跟0或多个属性。STUN头部包含一STUN消息类型、magic cookie、事务ID和消息长度。

0	1	2	3
00	STUN消息类型	消息长度	
魔术字			
事务ID（96位）			

每个STUN消息的最高位前2位必须为0。当STUN协议为多个协议多路复用时若使用的是同一个端口，这可以用于与其他协议区分STUN数据包。

消息类型确定消息的类别（如请求、成功回应、失败回应、标志）。虽然这里有四种消息类型，但可以分为2类事务：请求/响应事务、标志事务。

消息类型字段可进一步划分为下面结构：

M11	M10	M9	M8	M7	C1	M6	M5	M4	C0	M3	M2	M1	M0

消息类型定义如下：

0b00，表示请求

0b01，表示标志

0b10，表示成功响应

0b11，表示错误响应

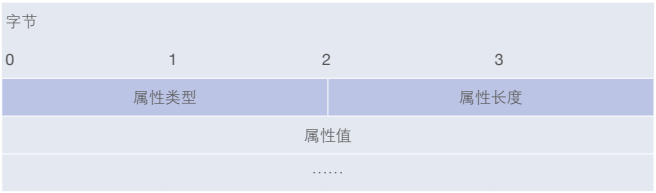
魔术字域必须包含固定的值0×2112A442。在RFC3489中，该域是事务ID的一部分。配置魔术字允许服务器检测客户是否理解某些在改进的版本中增加的属性。另外，还可用于STUN多路复用时与其他协议的包进行区分。

96位的事务ID用于唯一的识别STUN事务。对于请求/响应事务，事务ID由STUN客户端来选择；对于标志事务，由代理（代理指支持STUN的客户端或服务器）来选择并发送。它主要服务于与请求相关的响应，因此它也扮演着一个帮助阻止确定类型的攻击的角色。服务器使用事务ID来唯一的标识出所有客户端的每一个事务。事务ID本身必须是唯一的，并且随机的从0到2的96-1次方中选择。重新发送相同的请求时，也必须使用新的事务ID。成功或错误响应必须携带与相对应的请求相同的事务ID。

消息长度字段不包括20字节的STUN头部。所有的STUN属性必须填充为4字节的倍数。消息长度字段的最后2位总是为0，这为区分STUN包与其他协议的包提供了另外一种方法。

消息属性

STUN头之后是0或多个属性。每个属性都采用TLV编码，16位的类型、16位的长度及可变长度的值。每个STUN属性必须是4字节边界对齐。



属性空间被划分为2个范围。属性的类型值在0×0000到0×7fff是强制理解属性，这意味着除非STUN代理能够理解这些属性，否则将不能正常处理包含该属性的消息；属性的类型值在0×8000到0×ffff范围是可选理解属性，这意味着如果STUN代理不能理解它们的话这些属性可以被忽略。

STUN属性类型集由IANA维护。

■ MAPPED-ADDRESS

MAPPED-ADDRESS属性标识了客户端反向传输地址（映射后的地址），这个属性只用于服务器向后兼容RFC3489的客户端。

■ XOR-MAPPED-ADDRESS

XOR-MAPPED-ADDRESS属性与MAPPED-ADDRESS属性是相同的，除了这映射后的地址经过了异或处理。（注意，异或运算是其自身的逆运算，再异或一下就可以得出真实的MAPPED-ADDRESS）。

■ USERNAME

username属性用于消息完整性。它采用username和password组合方式用于消息完整性检查。

■ MESSAGE-INTEGRITY

MESSAGE-INTEGRITY属性包含STUN消息的HMAC-SHA1。它可以出现在任何类型的STUN消息中。由于使用SHA1散列算法，HMAC将会是20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。除了FINGERPRINT属性外，代理必须忽略其他出现在MESSAGE-INTEGRITY属性后的任何属性。

STUN消息头中的长度字段的值必须包括直到MESSAGE-INTEGRITY属性本身，但不包括任何在它之后的属性。

■ FINGERPRINT

FINGERPRINT属性可以存在于所有的STUN消息中，提供辅助区分STUN数据包与其他协议数据包的功能。属性的值为采用CRC32方式计算STUN消息直到但不包括FINGERPRINT属性的结果，并与32位的值0×5354554e异或。

■ ERROR-CODE

ERROR-CODE属性被用于错误响应消息中。它包含一个在300至699范围内的错误响应号。错误响应号定义如下：

300：尝试代替，客户端应该使用该请求联系一个代替的服务器。这个错误响应仅在请求包括一个username属性和一个有效的MESSAGE-INTEGRITY属性时发送；否则它不会被发送，而是发送错误代码为400的错误响应。

400：错误请求，请求是变形了，客户在修改先前的尝试前不应该重试该请求。

401：未授权，请求未包括正确的资格来继续。客户应该采用一个合适的资格来重试该请求。

420：未知属性，服务器收到一个STUN包包含一个强制理解的属性但是它不会理解。服务器必须将不认识的属性放在错误响应的UNKNOWN-ATTRIBUTE属性中。

438：过期Nonce，客户使用的Nonce不再有效，客户应该使用响应中提供的Nonce来重试。

500：服务器错误，服务器遇到临时错误，客户应该再次尝试。

■ REALM

REALM属性可能出现在请求和响应中。在请求中表示长期资格将在认证中使用。当在错误响应中出现表示服务器希望客户使用长期资格来进行认证。

■ NONCE

NONCE属性可能出现在请求和响应消息中。

■ UNKNOWN-ATTRIBUTES

UNKNOWN-ATTRIBUTES属性只在错误代码为420的错误响应中出现。

■ SOFTWARE

SOFTWARE属性用于代理发送消息时包含版本的描述。它用于客户端和服务器。它的值包括制造商和版本号。该属性对于协议的运行没有任何影响，仅为诊断和调试目的提供服务。SOFTWARE属性是个可变长度的，采用UTF-8编码的小于128个字符的序列号。

■ ALTERNATE-Server

ALTERNATE-Server属性标识一个备份的传输地址表明一个STUN客户可以尝试不同的STUN服务器。属性格式与MAPPED-ADDRESS相同。IP地址族必须与请求的源IP地址的相同。

RFC5389与RFC3489的区别

RFC5389与RFC3489的不同点如下：

■ 去掉STUN是一种完整的NAT穿透方案的概念，现在是一种用于提供NAT穿透解决方案的工具。因而，协议的名称变为NAT会话穿透效用；

■ 定义了STUN的用途；

■ 去掉了STUN关于NAT类型检测和绑定生命期发现的用法，去掉了RESPONSE-ADDRESS、CHANGED-ADDRESS、CHANGE-REQUEST、SOURCE-ADDRESS和REFLECTED-FROM属性；

■ 增加了一个固定的32位的魔术字字段，事务ID字段减少了32位长度；

■ 增加了XOR-MAPPED-ADDRESS属性，若魔术字在捆绑请求中出现时，该属性包括在捆绑响应中。否则，RFC3489中的行为是保留的（换句话说，捆绑响应中包括MAPPED-ADDRESS）；

■ 介绍了消息类型字段的正式结构，带有一对明确的位来标识Request、Response、Error-Response或Indication消息。因此，消息类型字段被划分为类别和方法两部分；

■ 明确的指出了STUN的最高2位是0b00，当用于ICE时可以简单的与RTP包区分开来；

■ 增加指纹属性来提供一种明确的方法来检测当STUN协议多路复用时，STUN与其他协议之间的差异；

■ 增加支持IPv6，IPv4客户端可以获取一个IPv6映射地址，反之亦然；

■ 增加一个long-term-credential-based认证机制；

■ 增加了SOFTWARE、REALM、NONCE和ALTERNATE-Server属性；

■ 去掉了共享密匙方法，因此password属性也去掉了；

■ 去掉了使用连续10秒侦听STUN响应来识别一个攻击的做法；

■ 改变事务计时器来增加TCP友好性；

■ 去掉了STUN例子，如集中分离控制和媒体面，代替的，在使用STUN协议时提供了更多的信息；

■ 定义了一类填充机制来改变长度属性的说明；

■ REALM、Server、原因语句和NONCE限制在127个字符，username限制在513个字节以内；

■ 为TCP和TLS改变了DNS SRV规程，UDP仍然和以前保持一致。

新特性介绍

指纹机制

FINGERPRINT机制是一种可选的用于其他协议多路复用STUN时发送给相同的传输地址时区分STUN数据包的机制，该机制不支持与RFC3489相兼容。

在一些用途中，基于相同的传输地址时多个协议会多路复用STUN消息，例如RTP协议。STUN消息必须首先和应用报文分离开。目前，在STUN报头中有3种固定的字段可以用于该目的。尽管如此，在一些案例中，三种固定字段仍然不能充分的区别开。

当扩展的指纹机制被使用时，STUN代理在发送给其他STUN代理的消息中包括FINGERPRINT属性。当其他STUN代理收到时，除基本的检查之外，还将检查是否包含FINGERPRINT属性及它是否包含正确的值，至此，它将相信这是一个STUN消息。指纹机制帮助STUN代理检查其他协议那些看起来像是STUN消息的消息。

通过DNS发现服务器机制

STUN客户端可以使用DNS来发现STUN服务器的IP地址和端口。客户端必须知道服务器的域名。

当客户端希望找出服务器在公网上的位置就采用捆绑请求/响应事务，SRV（资源记录表）中服务器名称是“stun”。当通过TLS会话采用捆绑请求/响应事务，SRV中服务器名称为“stuns”。STUN用户可以定义额外的DNS资源记录服务名称。

STUN请求的默认端口是3478，用于TCP和UDP。STUN在TLS上的默认端口是5349。服务器能够在TLS上运行STUN与STUN在TCP上时使用相同的端口，只有服务器软件支持决定初始消息是否是TLS或STUN消息。

如果SRV中没有记录可查，客户端执行A或AAAA记录查找域名。结果将会是1张IP地址表，每一个都可以使用TCP或UDP采用默认端口号连接。通常要求使用TLS，客户端使用STUN在TLS上的默认端口号连接其中一个IP地址。

认证和消息完整性机制

短期证书机制

短期证书机制假设在STUN事务之前，客户端和服务器已经使用了其他协议来交换了证书，以username和password形式。这个证书是有时间限制的。

例如，在ICE用途中，两个终端使用带外方式交换信息来对username和password达成一致，并在媒体会话期间使用。

这个证书被用来进行消息完整性检查，用于每个请求和多个响应中。与长期证书机制相比，没有挑战和响应方式，因此，这种证书的时间限制特性的优点是可以阻止重播。

长期证书机制

长期证书机制依赖于一个长期证书，username和password在客户端和服务器中是共用的。这个证书从它提供给用户开始将一直是有效的，直到该用户不再是该系统的用户。

这本质上是一个提供给用户username和password的传统的登入方式。

客户端初始发送一个请求，没有提供任何证书和任何完整性检测。服务器拒绝这个请求，并提供给用户一个范围（用于指导用

户或代理选择username和password）和一个nonce。这个nonce提供重放保护。它是一个cookie，由服务器选择，以这样一种方式来标示有效时间或客户端身份是有效的。客户端重试这个请求，这次包括它的USERNAME和REALM和服务器提供的nonce来回应。服务器确认这个nonce和检查这个message integrity。如果它们匹配，请求则通过认证。如果这个nonce不再有效，即过期了，服务器就拒绝该请求，并提供一个新的nonce。

在随后得到同一服务器的请求，客户端重新使用这个nonce、USERNAME和REALM，和先前使用的password。这样，随后的请求不会被拒绝直到这个nonce变成无效的。

需要注意的是，长期证书机制不能用来保护Indications，由于Indications不能被改变，因此，使用Indications时要么使用短期证书，要么就省略认证和消息完整性。

因为长期证书机制对离线字典攻击敏感，部署的时候应该使用很难猜测的密码。

备份服务器机制

服务器使用增强的重定向功能将一个客户端转向另一个服务器，通过回应一个错误响应号为300（尝试备份）的错误响应。服务器在错误响应中携带一个ALTERNATE-Server属性。

客户端收到错误响应号为300的错误响应后，在该响应中查找ALTERNATE-Server属性。若找到一个，客户端就会将当前的事务作废，并重新尝试发送请求到该属性中列出的服务器。请求报文若已经通过认证，则必须使用与先前发送给执行重定向操作的服务器同样的证书。如果客户端在最后5分钟里已经重试发送请求时已经重定向到了一个服务器，它必须忽略重定向操作并将当前的事务作废，这是为了防止无限的重定向循环。

RFC5389与RFC3489的兼容

在RFC3489中：

- UDP是唯一支持的传输协议；
- RFC5389中的魔术字字段是RFC3489中事务ID的一部分，事务ID长128位；
- 没有XOR-MAPPED-ADDRESS属性，绑定方法是使用MAPPED-ADDRESS属性代替；
- 有3个需要强制理解的属性，分别是：RESPONSE-ADDRESS、CHANGE-REQUEST、CHANGED-ADDRESS属性，而RFC5389中不再支持这些属性。

客户端处理的改变

客户端想要与RFC3489的服务器互操作，应发送一个使用绑定方法的请求消息，不包含任何消息，使用UDP协议发送给服务器。如果成功，将收到服务器发回的包含MAPPED-ADDRESS属性而不是XOR-MAPPED-ADDRESS属性的成功响应。客户端试图与基于RFC3489的应用服务器互操作必须准备好接收任意一个属性。此外，客户端必须忽略任何在响应中出现的保留的强制理解的属性。RFC3489中规定保留属性中的0×0002、0×0004、0×0005和0×000B可能出现在绑定响应中。

服务器处理的改变

服务器能够察觉由RFC3489中的客户端发送的携带有不正确的魔术字的捆绑请求消息。当服务器察觉到RFC3489中的客户端，它应该将捆绑请求消息中魔术字域中的值拷贝到捆绑响应中的魔术字字段中，并且插入一个MAPPED-ADDRESS属性代替XOR-MAPPED-ADDRESS属性。

客户端在极少的环境下可能包括RESPONSE-ADDRESS或CHANGE-REQUEST属性中的一个。在这些情况下，服务器把这些属性看做是一个不认识的强制理解的属性，并回应一个错误响应。

RFC3489版本中的STUN缺少魔术字和指纹属性这两种能够高可靠性的正确标识其他协议多路复用时的STUN消息。因此，STUN执行与RFC3489兼容时不应该被用于多个协议。

TURN

RFC5766/TURN

TURN，在RFC5766中定义，英文全称Traversal Using Relays around NAT（TURN）：Relay Extensions to session Traversal Utilities for NAT（STUN），即使用中继穿透NAT：STUN的中继扩展。简单的说，TURN与STUN的共同点都是通过修改应用层中的私网地址达到NAT穿透的效果，异同点是TURN通过两方通讯的“中间人”方式实现穿透。

如果一个主机位于NAT的后面，在某些情况下它不能够与其他主机点对点直接连接。在这些情况下，它需要使用中间网点提供的中继连接服务。TURN协议就是用来允许主机控制中继的操作并且使用中继与对端交换数据。TURN与其他中继控制协议不同的是它能够允许一个客户端使用一个中继地址与多个对端连接。

TURN协议被设计为ICE的一部分，用于NAT穿越，虽然如此，它也可以在没有ICE的地方单独使用。

操作概述

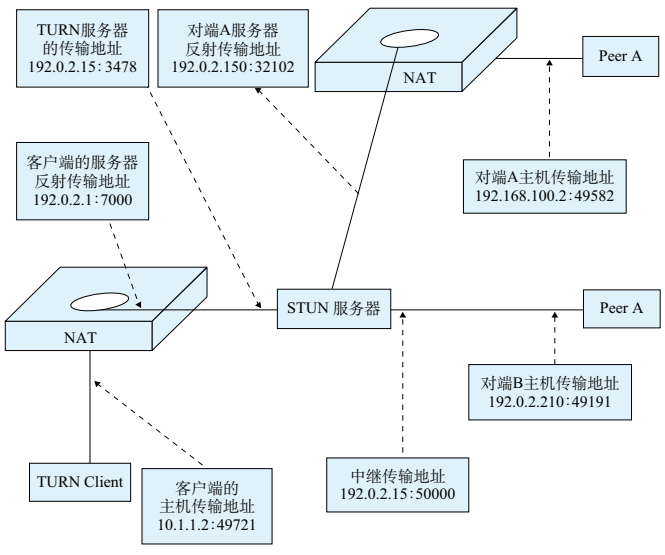


图2 TURN

在一个典型组网中，一个TURN客户端连接在一个私有网络中，通过一个或多个NAT来连接到公网。在公网中有一个TURN服务器。在因特网的别处有一个或多个对端是这个TURN客户端希望通讯的。这些对端也有可能是在一个或多个NAT的后面。该客户端使用服务器作为一个中继来发送数据包到这些对端去，并且从这些对端接收数据包。

客户端通过一个IP地址和端口的组合来与服务器建立会话。客户端使用TURN命令在服务器上创建和操作一个ALLOCATION。一旦这个ALLOCATION创建好了，客户端能够在数据发往哪个对端的指示下发送应用数据到这个服务器，服务器将中继这些数据到合适的对端。客户端发送的应用数据包含在TURN消息中，服务器将数据提取出来，并以UDP数据包方式发送给对端。反向上，对端以UDP数据包方式发送应用数据到这个ALLOCATION提供的中继传输地址。因为TURN消息总是包含客户端与哪些对端通讯的指示，客户端能够使用单一的ALLOCATION来与多个对端通讯。

术语

TURN client：遵循RFC5766的STUN客户端。

TURN Server：遵循RFC5766的STUN服务器。

Peer：TURN客户端希望连接的主机。TURN服务器为TURN客户端和它的对端中继流量，但Peer并不与TURN服务器使用TURN协议进行交互，它接收从TURN服务器发送过来的数据，并向TURN服务器发送数据。

Transport Address：IP地址与端口号的组合。

Host Transport Address：客户端或对端的传输地址。

Server-Reflexive Transport Address：NAT公网侧的传输地址，该地址由NAT分配，相当于一个特定的主机传输地址。

Relayed Transport Address：TURN服务器上的传输地址，用于客户端和对端中继数据。

TURN Server Transport Address：TURN服务器上的传输地址，用于客户端发送STUN消息给服务器。

Peer Transport Address：服务器看到的对端的传输地址，当对端是在NAT后面，则是对端的服务器反射传输地址。

ALLOCATION：通过Allocate请求将中继传输地址提供给客户端，除了中继状态外，还有许可和超时定时器等。

5-tuple：五元组，包括客户端IP地址和端口，服务器IP地址和端口，传输协议（包括UDP、TCP、TLS）的组合。

Channel：通道号与对端传输地址的关联，一旦一个通道号与一个对端的传输地址绑定，客户端和服务器就能够利用带宽效应更大的通道数据消息来交换数据。

Permission：一个对端允许使用它的IP地址和传输协议来发送数据到TURN服务器，服务器只为从对端发来的并且匹配一个已经存在的许可的流量中继到相应的客户端。

REALM：服务器内用于描述服务器或内容的一个字符串，这个REALM告诉客户端哪些用户名和密码的组合可用于认证请求。

Nonce：服务器随机选择的一个字符串，包含在报文摘要中。为了防止中继攻击，服务器应该有规律的改变这个nonce。

新的STUN方法

下面给出了新的STUN方法的编号：

0×003：Allocate
0×004：Refresh
0×006：Send
0×007：Data
0×008：CreatePermission
0×009：ChannelBind

新的STUN属性

0×000c：CHANNEL-NUMBER
0×000D：LIFETIME
0×0010：Reserved（was BANDWIDTH）
0×0012：XOR-PEER-ADDRESS
0×0013：DATA
0×0016：XOR-RELAYED-ADDRESS
0×0018：EVEN-PORT
0×0019：REQUESTED-TRANSPORT
0×001A：DON'T-FRAGMENT
0×0021：Reserved（was TIMER-VAL）
0×0022：RESERVATION-TOKEN

上面属性中的部分属性长度不是4字节的倍数，采用STUN的规则，使用1~3个padding字节来补齐。

■ CHANNEL-NUMBER

CHANNEL-NUMBER属性包含通道的号码。属性长4字节，包含16比特的无符号整数和2字节的RFFU（Reserved For Future Use）字段，该字段必须设为0且在接收时被忽略。

字节			
0	1	2	3
Channel Number		RFFU = 0	

■ LIFETIME

LIFETIME属性表示服务器在没有收到Refresh时维持一个ALLOCATION的持续时间。属性长4字节，包含一个32比特的无符号整数值，表示剩余多少秒终止。

■ XOR-PEER-ADDRESS

XOR-PEER-ADDRESS指定从TURN服务器看到的对端的地址和端口，例如如果对端是在一个NAT后面，则为对端的Server-reflexive传输地址。

■ DATA

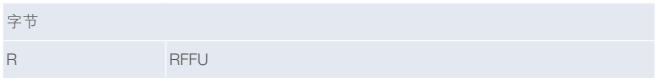
DATA属性存在于所有的Send和Data indications消息中。属性的值是可变长度的，包括应用数据。如果属性的长度不上4字节的倍数，必须进行填充。

■ XOR-RELAYED-ADDRESS

XOR-RELAYED-ADDRESS存在于所有的Allocate响应中。它指定了服务器分配给客户端的地址和端口。

■ EVEN-PORT

这个属性允许客户端请求在中继传输地址的端口为偶数，并且服务器可选的保留紧跟着的下一个端口号。属性的值长1字节，结构如下：

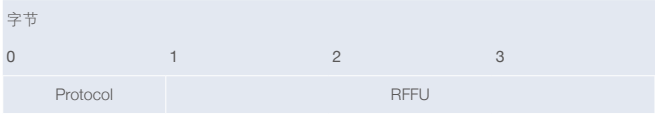


值包括一个1比特标志字段：

R：如果为1，服务器被请求保留下一个更高的端口号（基于同一个IP地址）为随后的ALLOCATION。如果为0，则不请求保留。属性的其他7比特值必须设置为0，并且在接收时被忽略。因为属性不是4字节的倍数，必须进行填充。

■ REQUESTED-TRANSPORT

客户端通过该属性为已分配的传输地址请求一个特定的传输协议。属性的值是4字节长度的。



协议字段指定了需求的协议。可以取自IPv4报头中的协议字段的值或IPv6报头的下一个报头字段的协议号。目前仅允许设置为17，即UDP。RFFU字段在传输时必须设置为0，并在接收时被忽略。保留用于未来使用。

■ DON'T-FRAGMENT

客户端使用该属性来请求服务器设置IP报头中的DF（不要分片）位，当中继应用数据到对端时。该属性没有值，因此属性长度字段为0。

■ RESERVATION-TOKEN

RESERVATION-TOKEN属性包含一个token来唯一的标识一个中继传输地址已经被服务器保留。服务器在一个成功响应中包含该属性来告诉客户端这个token，客户端在接下来的Allocate请求中包括该属性来请求服务器为这个ALLOCATION使用那个中继传输地址。属性值是8字节长。

新的STUN错误响应号

403，（Forbidden）：请求是有效的，但因管理或类似的规定而不能被执行。

437，（ALLOCATION Mismatch）：服务器接收到请求，要求在适当的位置的ALLOCATION，但没有ALLOCATION存在，或者一个收到的请求不指定任何ALLOCATION，但是一个ALLOCATION存在。

441，（Wrong Credentials）：请求中的证书没有匹配那些用来创建ALLOCATION的证书。

442，（不支持的传输协议）：Allocate请求要求服务器使用一个用于服务器和对端的传输协议，但是该服务器不支持该传输协议。

486，（ALLOCATION Quota Reached）：目前没有更多的ALLOCATIONs资源使用相同的用户名可被创建。

508，（Insufficient Capacity）：服务器不能够完成请求因为一些性能限制已经达到上限。在一个Allocate响应中，这可能因为服务器此时已经没有更多的中继传输地址资源了，没有更多的被请求的性能，或者相当于特定的保留的token不可用。

协议交互过程详细举例

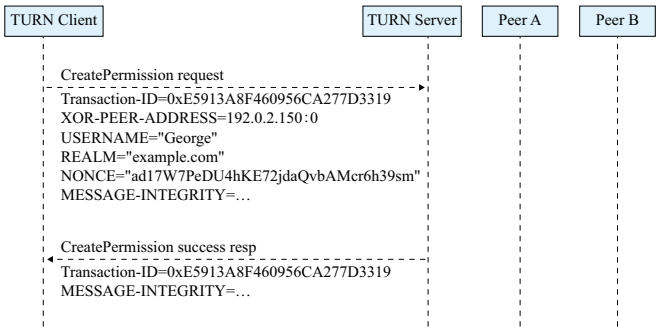
以图2为例进行讲解，每个消息中，多个属性包含在消息中并显示它们的值。为了方便阅读，值以人们可读的格式来显示。

客户端使用10.1.1.2:49271作为传输地址向服务器的传输地址发送Allocate请求。客户端随机选择一个96位的事务ID。该Allocate请求消息包括SOFTWARE属性来提供客户端的软件版本信息；包括LIFETIME属性，指明客户端希望该ALLOCATION具有1小时的生命期而非缺省的10分钟；包括REQUESTED-TRANSPORT属性来告诉服务器与对端之间采用UDP协议来传输；包括DON'T-FRAGMENT属性因为客户端希望在随后的Send indications中使用DON'T-FRAGMENT属性。

服务器需要任何请求必须是经过认证的，因此服务器拒绝了该最初的ALLOCATION请求，并且回应了携带有错误响应号为401（未授权）的Allocate错误响应；该响应包括一个REALM属性，指明认证的域；还包括一个NONCE属性和一个SOFTWARE属性。

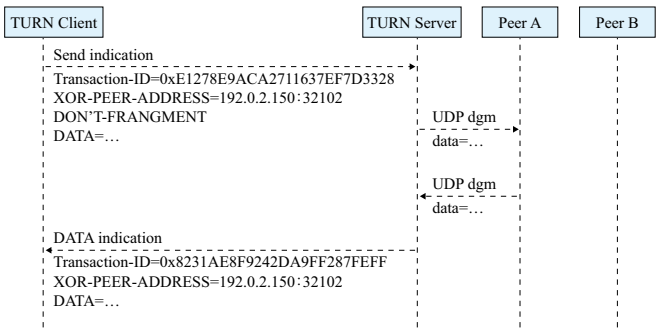
客户端收到了错误响应号为401的Allocate错误响应，将重新尝试发送Allocate请求，此时将包括认证属性。客户端在新的请求中重新选择一个新的事务ID。客户端包括一个username属性，使用从服务器那收到的REALM值来帮助它决定使用哪个值；请求还包括REALM和NONCE属性，这两个属性是从收到的错误响应中拷贝出来的。最后，客户端包括一个MESSAGE-INTEGRITY属性。

服务器收到认证的Allocate请求后，检查每个属性是否正确；然后，产生一个ALLOCATION，并给客户端回应Allocate成功响应。服务器在该成功响应中携带一个LIFETIME属性，本例中服务器将客户端请求的1小时生命期减小为20分钟，这是因为这个特定的服务器可能不允许超过20分钟的生命期；该响应包括XOR-RELAYED-ADDRESS属性，值为该ALLOCATION的中继传输地址；该响应还包括XOR-MAPPED-ADDRESS属性，值为客户端的Server-reflexive地址；该响应也包含一个SOFTWARE属性；最后，包括一个MESSAGE-INTEGRITY属性来证明该响应，确保它的完整性。



接着，客户端为了准备向对端A发送一些应用数据而创建一个permission。这里通过一个CreatePermission请求来做到。该请求携带XOR-PEER-ADDRESS属性包含有确定的请求的IP地址，这里为对端A的地址；需要注意的是，属性中地址的端口号被设置为0在CreatePermission请求中，并且客户端使用的是对端A的Server-reflexive地址而不是它的主机地址（私网地址）；客户端在该请求中携带与之前的Allocate请求中一样的USERNAME、REALM和NONCE值，因此该请求被服务器认可。此时在该请求中，客户端没有携带SOFTWARE属性。

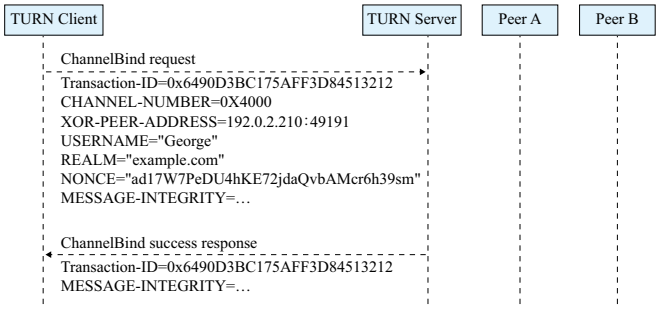
服务器收到该CreatePermission请求，产生一个相应的许可，并以CreatePermission成功响应来回应。该响应中只包含了Transaction-ID和MESSAGE-INTEGRITY属性。



现在客户端使用Send indication来发送应用数据到对端A。对端的Server-reflexive传输地址包含在XOR-PEER-ADDRESS属性中，应用数据包含在DATA属性中。客户端已经在应用层上执行了路径MTU发现功能，因此通过DON'T-FRAGMENT属性来告知服务器当通过UDP方式来向对端发送数据时应设置DF位。Indications不能使用长期证书机制来认证，所以该消息中没有MESSAGE-INTEGRITY属性。

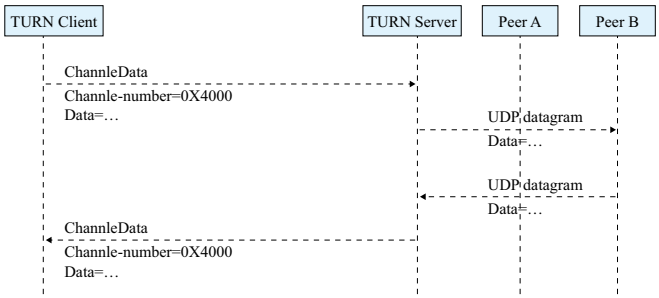
服务器收到Send indication后，提取出应用数据封装成UDP格式发给对端A；UDP报文的源传输地址为中继传输地址，并设置DF位。

对端A回应它自己的包含有应用数据的UDP包给服务器。目的地址为服务器的中继传输地址。当服务器收到后，将生成Data indication消息给客户端，携带有XOR-PEER-ADDRESS属性。应用数据包含在DATA属性中。



客户端现在若要绑定一个通道到对端B，将指定一个空闲的通道号（本例中为0×4000）包含在CHANNEL-NUMBER属性中，对端B的传输地址包含在XOR-PEER-ADDRESS属性中。与以前一样，客户端再次利用上次请求中的USERNAME、REALM和NONCE。

当服务器收到该请求后，服务器绑定这个对端的通道号，为对端B的IP地址安装一个permission，然后给客户端回应一个ChannelBind成功响应消息。



客户端现在发送一个ChannelData消息给服务器，携带有发送给对端B的数据。这个消息不是一个STUN消息，因此没有事务ID。它只有3个字段：通道号、数据、数据长度；服务器收到后，检查通道号后发现当前已经绑定了，就以UDP方式发送数据给对端B。

接着，对端B发送UDP数据包回应给服务器的中继传输地址。服务器收到后，回应给客户端ChannelData消息，包含UDP数据包中的数据。服务器知道是给哪个客户端发送ChannelData消息，这是因为收到的UDP数据包中的目的地址（即服务器的中继传输地址），并且知道使用的是哪个通道号，这是因为通道已经与相应的传输地址绑定了。

有时候，20分钟的生命期已经到了，客户端需要刷新ALLOCATION。此时通过发送Refresh请求来进行。该请求包含最后一次使用的USERNAME、REALM和NONCE，还包含SOFTWARE属性。当服务器收到这个Refresh请求时，它注意到这个nonce值已经超期了，则给客户端回应一个错误响应号为438（过期Nonce）的Refresh错误响应，并提供一个新的nonce值。客户端将重试该请求，此时携带新的nonce值。若第二次尝试被接受，服务器将回应一个成功响应。需要注意的是，此时客户端在请求中没有携带LIFETIME属性，所以服务器刷新客户端的ALLOCATION时采用缺省的10分钟生命期。

总结

在现实Internet网络环境中，大多数计算机主机都位于防火墙或NAT之后，只有少部分主机能够直接接入Internet。很多时候，我们希望网络中的两台主机能够直接进行通信（即所谓的P2P通信），而不需要其它公共服务器的中转。由于主机可能位于防火墙或NAT之后，在进行P2P通信之前，我们需要进行检测以确认它们之间能否进行P2P通信以及如何通信。这种技术通常被称为NAT穿透（NAT Traversal）。

RFC3489中定义的STUN，即简单地用UDP穿过NAT（STUN）是个轻量级的协议。它允许应用发现它们与公共互联网之间存在的NAT和防火墙及其他类型。它还为应用提供判断NAT给它们分配的公共网际协议（IP）地址。STUN可工作在许多现存NAT上，并且不需要它们做任何特别的行为。它允许广泛的各类应用穿越现存的NAT设施。

RFC5389中对STUN协议进行了修订，将其定位于为穿透NAT提供工具，即NAT会话穿透效用是一个用于其他解决NAT穿透问题协议的协议。它可以用于终端设备检查由NAT分配给终端的IP地址和端口号。同时，它也被用来检查两个终端之间的连接性，好比是一种维持NAT绑定表项的保活协议。STUN本身并不是一种完整的NAT穿透解决方案。它相当于是一种NAT穿透解决方案中的工具。这是与先前的版本相比最重要的改变。之前的RFC3489中定义的STUN是一个完整的穿透NAT解决方案。此外，最大的区别是支持TCP穿透。

RFC5766中对STUN协议再次进行了扩展，即中继穿透NAT：STUN的扩展。TURN与STUN的共同点都是通过修改应用层中的私网地址达到NAT穿透的效用，异同点是TUN采用了两方通讯的“中间人”方式实现穿透，突破了原先STUN协议无法在两台主机不能够点对点直接连接下提供作用的限制。

技术无止境，NAT穿透技术仍在不断更新中，这里只对STUN/TURN协议作了简单的介绍，具体细节请参考RFC3489/5389/5766。

Punching的实现与应用

文/林鹏程

Hole Punching技术是一种借助于公网上的服务器完成NAT穿越的解决方案，一开始单为解决UDP的NAT穿越而提出，但是在某些条件（主要是操作系统支持）满足的情况下，Hole Punching技术同样适用于TCP的NAT穿越。相对其他NAT穿越方案来说，Hole Punching技术显得比较通用和简单。

UDP Hole Punching

在基于UDP的Hole Punching场景中，终端A，B分别与公网上的服务器S建立UDP连接。当一个终端向服务器S注册时，服务器S记录下该终端的两对IP地址和端口信息，为描述方便，我们将一对IP地址和端口信息的组合称之为一个endpoint。一个endpoint是终端发起与服务器S通信的IP地址和端口；另一个endpoint是服务器S观察到的该终端实际与自己通信所用的IP地址和端口。我们可以把前一个endpoint看作是终端的私网IP地址和私网端口；把后一个endpoint看作是终端的私网IP地址和端口经过NAT转换后的公网IP地址和公网端口。服务器S可以从终端的注册报文中得到该终端的私网endpoint相关信息，可以通过对注册报文的源IP地址和UDP源端口字段获得该终端的公网endpoint。如果终端不是位于NAT设备后面，那么采用上述方法得到的两个endpoint应该是完全相同的。

也有一些NAT设备会扫描UDP报文的数据字段，寻找4字节的位域，将看上去很像IP地址的位域，改为与IP头一样的地址。为了避免这种行为的NAT设备对UDP报文数据的修改，应用程序可以采用直接对IP地址的值进行加密的方式骗过NAT设备的检查。

当终端A希望与终端B建立连接时，Hole Punching过程如下所示：

■ 终端A最初并不知道如何向B发起连接，于是终端A向服务器S发送报文，请求服务器S帮助建立与终端B的UDP连接；

■ 服务器S将含有终端B的公网及私网的endpoint发给终端A，同时，服务器S将含有终端A的公网及私网的endpoint的用于请求连接的报文也发给B。一旦这些报文都顺利达到，终端A与终端B都知道了对方的公网和私网的endpoint；

■ 当终端A收到服务器S返回的包含终端B的公网和私网的endpoint的报文后，终端A开始分别向这些终端B的endpoint发送UDP报文，并且终端A会自动“锁定”第一个给出响应的终端B的endpoint。同理，当终端B收到服务器S发送的包含终端A的公网和私网的endpoint的报文后，也会开始向终端A的公网和私网的endpoint发送UDP报文，并且自动锁定第一个得到终端A的回应的endpoint。由于终端A与B的互相向对方发送UDP报文的操作是异步的，所以终端A与B发送报文的时间先后没有严格的时序要求。

下面我们就来看一下这三个角色之间是如果进行UDP Hole Punching的。在这里，我们分为三种具体情景来讨论：第一种也是最“简单”的一种情景，两个终端都位于同一个NAT设备后面，位于同一个内网中；第二种也是最普遍的一种情景，两个终端分别位于不同的NAT设备后面，分属不同的内网；第三种是终端位于两层NAT设备之后，通常最上层的NAT是由ISP提供，第二层的NAT是家用的NAT路由器之类的设备。

通常情况下由应用程序自身确定的网络物理层连接方式是很困难的，有时甚至是不可能的，即使是上述的若干种情景下可以穿越NAT，也只是代表在一定时期内有效，而不是永久有效的。诸如STUN之类的网络协议或许可以提供必要的NAT信息，但在遇到多层NAT设备的时候，通常这些信息也不是完全完整和有效的。尽管如此，只要NAT设备的响应是“合理”的，在通常情况下Hole Punching技术还是能够在应用程序对网络状况一无所知的前提下自动适用于多数场合。（“合理”的NAT响应将在后文“P2P友好NAT”部分中详细讨论）

终端位于同一个NAT设备后面

首先假设两个终端位于同一个NAT设备后面，并且位于相同的内网（相同的私有IP地址域）如图1~3所示。终端A与服务器S建立了UDP连接，经过NAT转换后，终端A的公网端口被映射为2000。终端B同样与服务器S建立了UDP连接，公网端口映射为2010。

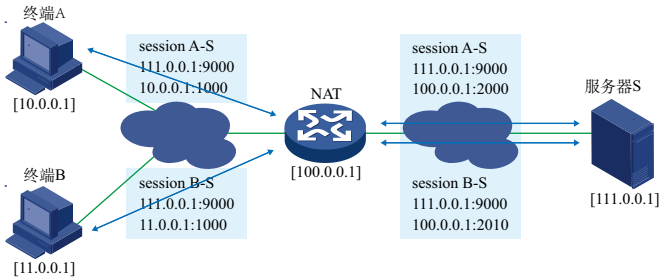


图1 终端位于同一个NAT后面图，Hole Punching前

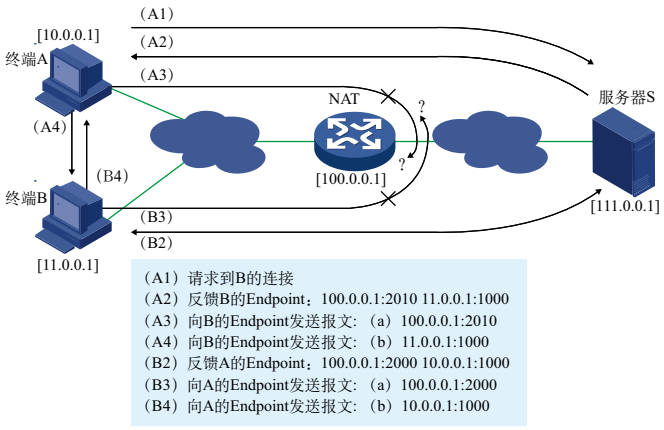


图2 终端位于同一个NAT后面图，Hole Punching时

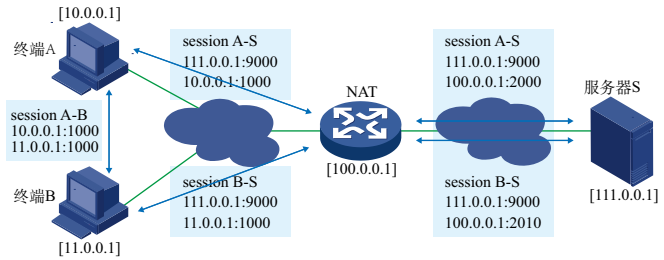


图3 终端位于同一个NAT后面图，Hole Punching后

假设终端A想通过服务器S，利用Hole Punching技术与终端B建立连接。终端A向服务器S发出请求报文与终端B进行连接。服务器S将终端B的公网和私网endpoint信息发给终端A，同时也把终

端A的公网和私网的endpoint发给终端B。由终端A和B发往对方公网endpoint的UDP报文能否被对方收到，这取决于当前的NAT设备是否支持回环转换（hairpin translation，详见后文“终端位于多层NAT设备后面”部分）。但是终端A与B往对端私网endpoint发送的UDP报文是一定可以到达的，无论如何，私网报文采用最短转发路径，要比经过NAT转换来的快。终端A与B有很大的可能性采用私网的endpoint进行常规的通信。

假定NAT设备支持回环转换，应用程序也忽略私网endpoint间的连接，那么终端A、B会采用公网endpoint作为通信的连接，这势必会造成数据报文不必要的经过NAT设备，这是一种对资源的浪费。就目前的网络情况而言，应用程序最好还是把公网和私网endpoint都实验一下。

终端位于不同的NAT设备后面

假定终端A与B在不同的NAT设备后面，分属不同的内网，如图4~6所示。终端A与B都经由各自的NAT设备与服务器S建立了UDP连接，A与B的本地私网端口号均为1000，服务器S的公网端口号为9000。在NAT设备的映射关系中，终端A的公网IP被映射为100.0.0.1，公网端口为2000，终端B的公网IP被映射为101.0.0.1，公网端口为2000。

如下所示：终端A——>本地私网IP：10.0.0.1，本地私网端口:1000，公网IP：100.0.0.1，公网端口：2000，终端B——>本地私网IP：11.0.0.1，本地私网端口:1000，公网IP：101.0.0.1，公网端口：2000。

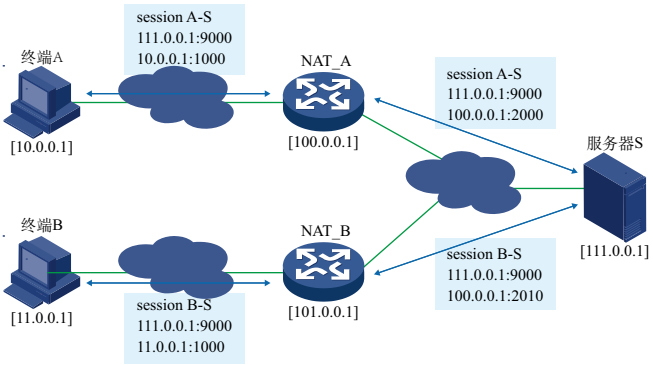


图4 终端位于不同的NAT后面图，Hole Punching前

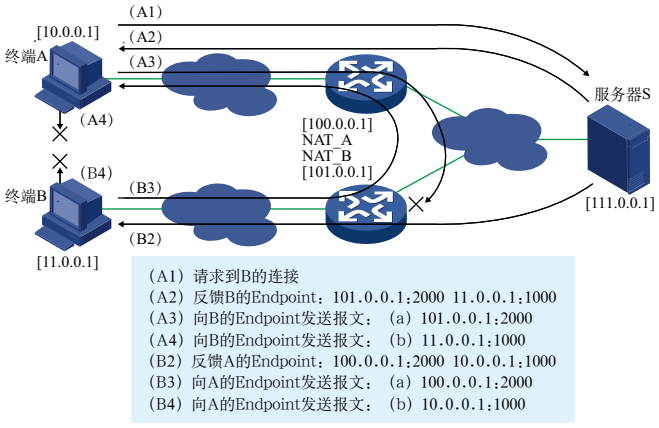


图5 终端位于不同的NAT后面图，Hole Punching时

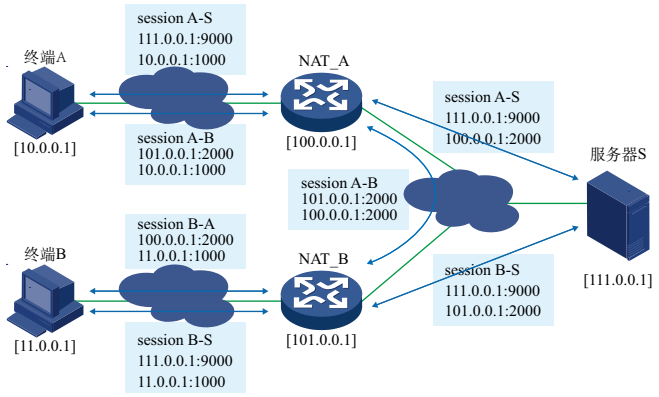


图6 终端位于不同的NAT后面图，Hole Punching后

在终端A向服务器S发送的注册报文中，会包含终端A的私网endpoint信息，即10.0.0.1:1000；服务器S会记录下终端A的私网endpoint信息，同时会把自己观察到的终端A的公网endpoint记录下来，即100.0.0.1:2000。同理，服务器S会记录下终端B的私网endpoint信息，11.0.0.1:1000和由服务器S观察到的终端B的公网endpoint，101.0.0.1:2000。无论终端A与B二者任何一方向服务器S发送连接请求，服务器都会将其记录下来的上述的公网、私网endpoint信息发送给对方。

由于终端A、B分属不同的内网，它们彼此的私网endpoint无法在公网中路由，所以发往各自私网endpoint的UDP报文会被发送到错误的主机或者根本不存在的主机。因此应用程序对于收到的报文必须经过授权和过滤。例如，可以在报文中加入对方的程序名

称、加密算法，或者至少是一个双方都从服务器S上预先得到的随机数字。

现在假定终端A的第一个报文将发往终端B的公网endpoint，如图5所示。该消息途经终端A的NAT设备，并在该设备上生成了一个映射。新的映射源endpoint是10.0.0.1:1000，该映射关系和终端A与服务器S的建立连接的时候NAT生成的映射关系的源endpoint是一样，但它的目的endpoint是不同。如果终端A的NAT设备给出的响应是“友好”的，那么终端A的NAT设备将保留终端A的私网endpoint，并且所有来自终端A的私网源endpoint（10.0.0.1:1000）的数据报文都沿用终端A与服务器S事先建立起来的映射关系，公网endpoint均为（100.0.0.1:2000）。终端A向终端B的公网endpoint发送消息的过程就是“打洞”的过程，从终端A的内网的角度来看应为从（10.0.0.1:1000）发往（101.0.0.1:2000），从终端A在其NAT设备上建立的映射来看，是从（100.0.0.1:2000）发到（101.0.0.1:2000）。

如果终端A发给终端B的公网endpoint的报文在终端B向终端A发送报文之前到达终端B的NAT设备，终端B的NAT会认为终端A发过来的报文是未经授权的公网报文，会丢弃掉该报文。终端B发往终端A的报文跟上述的过程一样，会在终端B的NAT上建立一个（11.0.0.1:1000，100.0.0.1:2000）的映射关系（通常也会沿用终端B与服务器S连接时建立的映射，只是该映射现在不光可以接受由服务器S发给终端B的报文，还可以接受从终端A的NAT设备-100.0.0.1:2000发来的报文）。

一旦终端A与终端B都向对方的NAT设备在公网上的endpoint发送了报文，就打开了终端A与终端B之间的“洞”，终端A与终端B向对方的公网endpoint发送数据报文，等效为向对方的客户端直接发送UDP数据报文了。一旦应用程序确认已经可以通过往对方的公网endpoint发送数据报文的方式让数据报文到达NAT设备后面的目的应用程序，程序会自动停止继续发送用于“Hole Punching”的数据报文，转而开始真正的数据通信。

终端位于多层NAT设备后面

有的网络拓扑结构包含了多个NAT设备，如果没有掌握该拓扑结构的详细信息，两个终端之间是无法建立“最优化”的点对点路由的。现在我们来讨论最后一种情况，如图7~9所示。假定NAT C是由ISP（Internet Service Provider）提供的工业级的NAT设备，NAT C提供将多个下属的用户NAT或用户节点映射到有限的几个公网IP的服务，NAT A和NAT B作为NAT C的内网节点将把用户的家庭网络或内部网络接入NAT C的内网，然后用户的内部网络就可以经由NAT C访问公网了。从这种拓扑结构上来看，只有服务器S与NAT C是真正拥有公网可路由IP地址的设备，而NAT A和NAT B所使用的“公网”IP地址，实际上是由ISP服务提供商设定的（相对于NAT C而言）私网地址（本文的后续部分，把这个由ISP提供的私网地址相对于NAT A和NAT B称之为“伪”公网地址），同理隶属于NAT A与NAT B的客户端，相对于NAT A、NAT B而言，它们处于NAT A、NAT B的内网，以此类推，客户端可以放到多层NAT设备后面。客户端A和客户端B发起对服务器S的连接的时候，就会依次在NAT A和NAT B上建立向外的映射关系，而NAT A、NAT B要联入公网的时候，会在NAT C上再建立向外的映射关系。

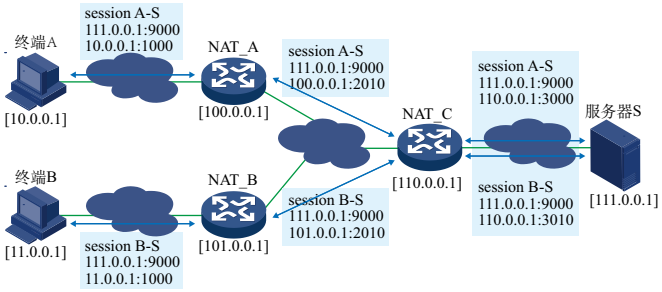


图7 终端位于多层NAT后面图，Hole Punching前

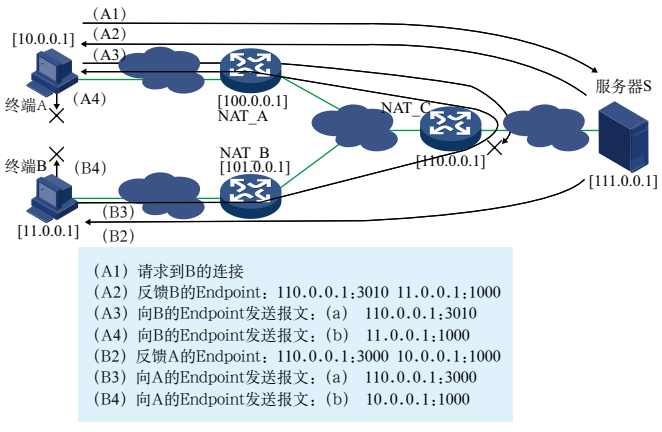


图8 终端位于多层NAT后面图，Hole Punching时

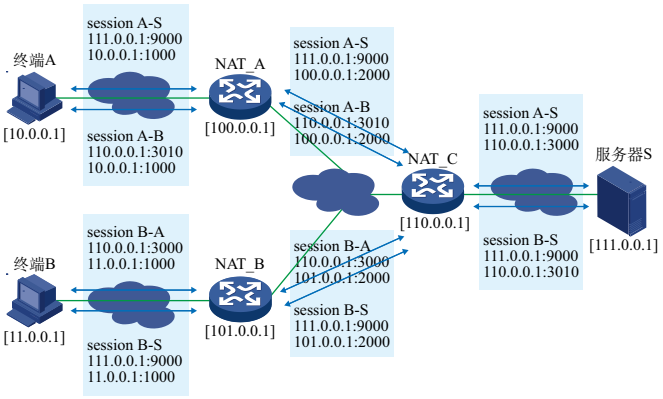


图9 终端位于多层NAT后面图，Hole Punching后

现在假定终端A和B希望通过UDP Hole Punching完成两个终端间的P2P直连。最优化的路由策略是终端A向终端B的“伪公网”IP上发送数据包，即ISP服务提供商指定的私网IP，NAT B的“伪”公网endpoint，101.0.0.1:2000。由于从服务器S的角度只能观察到真正的公网地址，也就是NAT A、NAT B在NAT C建立的映射时的真正的公网endpoint：110.0.0.1:3000以及110.0.0.1:3010，所以非常不幸，终端A与B是无法通过服务器S知道这些“伪”公网的endpoint。而且即使终端A和B通过某种手段可以得到NAT A和NAT B的“伪”公网endpoint，我们仍然不建议采用上述的“最优化”的打洞方式，这是因为这些地址是由ISP服务提供商提供的，或许会存在与客户端本身所在的私网地址重复的可能性。（例如：NAT A的私网IP地址域恰好与NAT A在NAT C的“伪”公网IP地址域重复，这样就会导致打洞报文无法发出的问题）

因此终端别无选择，只能使用由公网服务器S观察到的终端A，B的公网地址和端口进行“打洞”操作，用于“打洞”的报文将由NAT C进行转发，这里NAT C是否支持回环转换非常重要，否则数据包将无法由NAT C转发给NAT A和NAT B，进而无法到达终端A和B。当终端A向B的公网地址（110.0.0.1:3010）发送UDP数据报文的时候，NAT A首先把数据包的源地址由A的私网endpoint（10.0.0.1:1000）转换为“伪”公网endpoint（100.0.0.1:2000），现在报文到了NAT C，NAT C应该可以识别出来该数据包是要发往自身转换过的公网endpoint，如果NAT C可以给出“合理”响应的话，NAT C将把该数据包的源endpoint改为110.0.0.1:3000，目的endpoint改为101.0.0.1:2000，即NAT B的“伪”公网endpoint，NAT B最后会将收到的报文发往终端B。同样，由终端B发往A的数据包也会经过类似的过程。也有很多NAT设备不支持类似这样的回环转换，但是已经有越来越多的NAT设备生产厂商开始加入对该转换的支持。

UDP在空闲状态下的超时问题

由于UDP转换协议提供的“洞”不是绝对可靠的，多数NAT设备内部都有一个UDP转换的空闲状态计时器，如果在一段时间内没有UDP数据通信，NAT设备会关掉由“打洞”操作打出来的“洞”，作为应用程序来讲如果想要做到与设备无关，就最好在穿越NAT后设定一个穿越的有效期。这个有效期与NAT设备内部的配置有关，目前没有标准有效期，最短的只有20秒左右。在这个有效期内，即使没有P2P数据报文需要传输，应用程序为了维持该“洞”可以正常工作，也必须向对方发送“打洞”维持报文。这个维持报文是需要双方应用都发送的，只有一方发送不会维持另一方的映射关系正常工作。除了频繁发送“打洞”维持报文以外，还有一个方法就是在当前的“洞”有效期过期之前，P2P客户端双方重新“打洞”，丢弃原有的“洞”，这也不失为一个有效的方法。

TCP Hole Punching

建立穿越NAT设备的P2P的TCP连接只比UDP复杂一点点，TCP Hole Punching从协议层来看是与UDP Hole Punching过程非常相似的。尽管如此，基于TCP Hole Punching至今为止还没有被很好的理解，这也造成了对其提供支持的NAT设备不是很多。在NAT设备支持的前提下，基于TCP Hole Punching技术实际上与基于UDP Hole Punching技术一样快捷、可靠。实际上，只要NAT设备支持的话，基于TCP的P2P技术的健壮性将比基于UDP的技术的更强一些，因为TCP协议的状态机给出了一种标准的方法来精确的获取某个TCP映射关系的生命期，而UDP协议则无法做到这一点。

套接字和TCP端口的重用实现

基于TCP Hole Punching过程中，最主要的问题不是来自于TCP协议，而是来自于应用程序的API接口。这是由于标准的伯克利（Berkeley）套接字的API是围绕着构建客户端/服务器程序而设计

的，API允许TCP流套接字通过调用connect（）函数来建立向外的连接，或者通过listen（）和accept（）函数接受来自外部的连接，但是，API不提供类似UDP那样的，同一个端口既可以向外连接，又能够接受来自外部的连接。而且更糟的是，TCP的套接字通常仅允许建立1对1的响应，即应用程序在将一个套接字绑定到本地的一个端口以后，任何试图将第二个套接字绑定到该端口的操作都会失败。为了让TCP Hole Punching能够顺利工作，我们需要使用一个本地的TCP端口来监听来自外部的TCP连接，同时建立多个向外的TCP连接。幸运的是，所有的主流操作系统都能够支持特殊的TCP套接字参数，通常叫做“SO_REUSEADDR”，该参数允许应用程序将多个套接字绑定到本地的一个IP地址和端口（只要所有要绑定的套接字都设置了SO_REUSEADDR参数即可）。BSD（Berkeley Software Distribution，伯克利软件套件）系统引入了SO_REUSEPORT参数，该参数用于区分端口重用还是地址重用，在这样的系统里面，上述所有的参数必须都设置才行。

打开P2P的TCP流

假定终端A希望建立与终端B的TCP连接。我们像通常一样假定终端A和B已经与公网上的已知服务器S建立了TCP连接。服务器记录下来每个注册的客户端的公网和私网的endpoint，如同为UDP服务的时候一样。从协议层来看，TCP Hole Punching与UDP Hole Punching是几乎完全相同的过程。

■ 终端A使用其与服务器S的连接向服务器发送请求，要求服务器S协助其连接终端B；

■ 服务器S将终端B的公网和私网endpoint返回给终端A，同时，服务器S将终端A的公网和私网endpoint发送给终端B；

■ 客户端A和B使用连接服务器S的TCP端口异步地发起向对方的公网、私网endpoint的TCP连接，同时监听各自的本地TCP端口是否有外部的连接联入；

■ 终端A和B开始等待向外的连接是否成功，检查是否有新连接联入。如果向外的连接由于某种网络错误而失败，如：“连接被重置”或者“节点无法访问”，终端只需要延迟一小段时间（例如延迟一秒钟），然后重新发起连接即可，延迟的时间和重复连接的次数可以由应用程序编写者来确定；

■ TCP连接建立起来以后，终端之间应该开始鉴权操作，确保目前联入的连接就是所希望的连接。如果鉴权失败，终端将关闭连接，并且继续等待新的连接联入。终端通常采用“先入为主”的策略，只接受第一个通过鉴权操作的终端，然后将进入通信过程不再继续等待是否有新的连接联入。

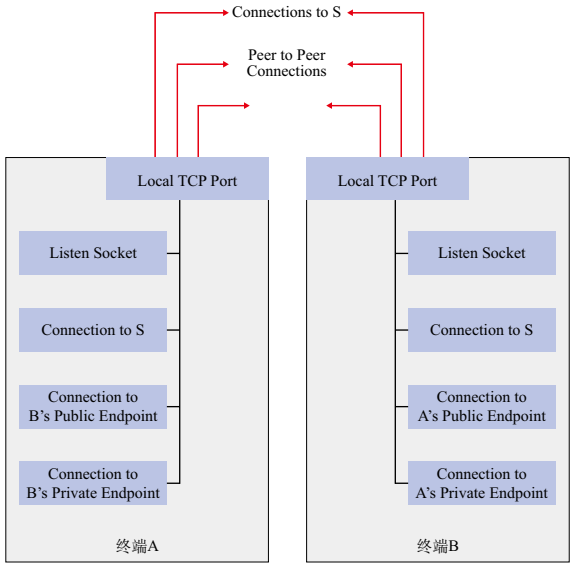


图10 TCP Hole Punching图

与UDP不同的是，使用UDP协议的每个终端只需要一个套接字即可完成与服务器S通信，并同时与多个P2P终端通信的任务，而TCP终端必须解决多个套接字绑定到同一个本地TCP端口的问题，如图10所示。现在来看更加实际的一种情景，终端A与B分别位于不同的NAT设备后面，如图5所示，并且假定图中的端口号是TCP协议的端口号，而不是UDP的端口号。图中向外的连接代表终端A和B向对方的私网endpoint发起的连接，这些连接或许会失败或者无法连接到对方。如同使用UDP Hole Punching遇到的问题一样，TCP Hole Punching也会遇到私网IP与“伪”公网IP重复造成连接失败或者错误连接之类的问题。客户端向彼此公网endpoint发起连接的操作，会使得各自的NAT设备打开新的“洞”允许终端A与B的TCP数据通过。如果NAT设备支持TCP“打洞”操作的话，一个在终端之间的基于TCP协议的流通道就会自动建立起来。如果终端A向B发送的第一个SYN报文发到了终端B的NAT设备，而终端B在此前没有向终端A发送SYN报文，终端B的NAT设备会丢弃这个报文，这会引起终端A的“连接失败”或“无法连接”问题。而此时，由于终端A已经向终端B

发送过SYN报文，终端B发往终端A的SYN报文将被看作是由终端A发往终端B的报文的回应的一部分，所以终端B发往终端A的SYN报文会顺利地通过终端A的NAT设备，到达终端A，从而建立起终端A与B的P2P连接。

从应用程序的角度来看TCP Hole Punching

从应用程序的角度来看，在进行TCP Hole Punching的时候都发生了什么呢？假定终端A首先向终端B发出SYN报文，该报文发往终端B的公网地址，并且被终端B的NAT设备丢弃，但是终端B发往终端A的公网地址的SYN报文则通过终端A的NAT到达了终端A，然后，会发生以下的两种结果中的一种，具体是哪一种取决于操作系统对TCP协议的实现：

■ 终端A的TCP实现会发现收到的SYN报文就是其发起连接并希望联入的终端B的SYN报文，通俗一点来说就是“说曹操，曹操到”的意思，本来终端A要去找终端B，结果终端B自己找上门来了。终端A的程序调用的异步connect（）函数将成功返回，终端A的listen（）等待从外部联入的函数将没有任何反映。此时，终端B联入终端A的操作在终端A的程序内部被理解为终端A联入终端B连接成功，并且终端A开始使用这个连接与B开始P2P通信。由于收到的SYN报文中不包含终端A需要的ACK数据，因此，终端A的TCP将用SYN-ACK报文回应终端B的公网endpoint，并且将使用先前终端A发向终端B的SYN报文一样的序列号。一旦终端B的TCP收到由终端A发来的SYN-ACK报文，则把自己的ACK报文发给终端A，然后两端建立起TCP连接。简单的说，第一种，就是即使终端A发往终端B的SYN报文被终端B的NAT丢弃了，但是由于终端B发往终端A的报文到达了终端A。结果是，终端A认为自己连接成功了，终端B也认为自己连接成功了，不管是谁成功了，总之连接是已经建立起来了。

■ 另外一种结果是，终端A的TCP实现没有像前文中所讲的那么“智能”，它没有发现现在联入的终端B就是自己希望联入的。就好比在机场接人，明明遇到了自己想要接的人却不认识，误认为是其它的人，安排别人给接走了，后来才知道是自己错过了机会，但是无论如何，人已经接到了任务已经完成了。然后，A通过常规的listen（）函数和accept（）函数得到与B的连接，而由A发起的向B的公网地址的连接会以失败告终。尽管终端A向B的连接失败，终端A仍然得到了终端B发起的向A的连接，等效于终端A与B之间已经联通，不管中间过程如何，终端A与B已经连接起来了，结果是终端A和B的基于TCP协议的P2P连接已经建立起来了。

第一种结果适用于基于BSD的操作系统对于TCP的实现，而第二种结果更加普遍一些，多数Linux和Windows系统都会按照第二种结果来处理。

TCP同时打开

假定各终端的TCP连接启动时间比较巧合，使得他们各自发送的SYN报文，在到达对方的NAT设备之前，对方的SYN报文都已经穿越NAT设备，并在NAT设备上生成TCP映射关系。在这种“幸运”的情况下，NAT设备不会拒绝SYN报文，双方的SYN报文都能通过终端间的NAT设备，到达对方。此时，终端会发现TCP连接同时打开，每个终端的TCP返回SYN-ACK报文，报文的SYN部分必须与之前发送的SYN报文一样，而ACK部分告知对方到达的SYN信息。

在这种情况下，应用程序的实现依赖于TCP的实现。如果双方终端按照前一节中所描述第二种行为执行，可能最终程序所有的异步connect（）呼叫都会失败，但程序依然会收到一个新的、有效地P2P TCP流套接字accept（）函数。 由于应用程序并不关心是否它最终收到的P2P TCP套接字的connect（）或accept（），会导致有效地流可在任何依据RFC793标准的TCP上传输。

有序的Hole Punching

在这种变化的TCP Hole Punching流程中，终端倾向于采用异步的方式进行连接，比如：

■ 终端A通过服务器S告知终端B，需要建立一个连接，同时终端A并没有侦听本地端口；

■ 终端B尝试向A发起连接，并在NAT设备打洞，但是由于超时、接收到终端A侧NAT设备回应的RST报文或者终端A返回的RST而导致连接失败；

■ 终端B关闭与服务器S的连接，并用该TCP端口进行侦听；

■ 服务器S依次也关闭与终端A的连接，终端A再尝试用原端口直接与终端B建立连接。

这种序贯程序尤其被用于安装Windows XP SP2操作系统之前的主机，这些主机不能正确处理TCP同时开启，或者TCP套接字不支持SO_REUSEADDR的参数。这种序贯程序存在更多的不确定性，可能会使通信启动更慢或不稳定。

P2P友好NAT

本节主要描述了能支持上文涉及的Hole Punching技术的关键特性，并不是所有当前的NAT能支持这些特性，虽然大部分可以，同时NAT设备商会因市场的需要，而提供能更好支持P2P的设备。

一致的地址转换

本文描述的Hole Punching技术只有当NAT设备始终将TCP或者UDP私网源endpoint一致的映射到一个相对应的公网endpoint时，才能运行正常。在RFC3489中，称该类NAT设备为cone类型NAT，即NAT设备强制将所有来自同一个私网endpoint映射到一个相同的公网endpoint。

考虑如上文图4－6描述的例子中，当终端A初始连接服务器S时，NAT A分配100.0.0.1:2000来映射A的私网endpoint：10.0.0.1:1000，当终端A尝试与终端B建立连接时，采用相同的本地私网endpoint作为源IP地址和端口，终端B的公网endpoint作为目的IP地址和端口。终端A需要依靠NAT A保留私网地址的标识，并需要重新使用存在的公网endpoint：100.0.0.1:2000，因为该endpoint也是终端B发报文给终端A的目的endpoint。

对于symmetric类型NAT设备，由于NAT上分配用来连接服务器和对端设备的端口不一致，而导致通信失败。大部分symmetric类型NAT设备，会采用公平地、可预期的方式给连续的映射关系分配端口。对于这种情况，可以采用一种变化的Hole Punching算法，通过预测NAT设备分配给P2P连接的公网端口，可以解决该类NAT设备的穿越问题。但是由于分配的端口是一个动态的目标，有时候也会有错误发生，比如已经被其他映射关系占用了预期的端口等。因为symmetric类型NAT设备并不比cone类型NAT设备更加安全，设备商为了支持P2P协议而更少采用symmetric类型NAT。

处理主动发起的TCP连接

当NAT设备公网侧收到一个SYN报文，而没有对应的映射关系时，NAT设备会悄悄的丢弃SYN报文，这很重要。一些NAT设备会采用返回TCP RST报文，甚至ICMP错误报告报文替代丢弃行为，这样会影响TCP Hole Punching流程，会导致终端花费更长时间完成打洞。

保持载荷独立

少数NAT设备会去扫描报文载荷，获取4byte类似IP地址的信息，然后用报文IP头中的地址信息去替换这4byte字段。这种不好的处理方式会使得通信不正确，应用程序可以简单的保护他们发送报文中的IP地址信息，比如将IP地址采用二进制反码表示。

回环转换

一些多层NAT应用中，需要回环转换使得Hole Punching能正常工作，如前文提到的多层NAT设备后面的应用必须支持回环转换。

P2P中的NAT穿越方案简介

文/王军

P2P简介

P2P即点对点通信，或称为对等联网，与传统的服务器客户端模式有着明显的区别，传统的服务器客户端模型如图2所示。P2P这一术语在不同的上下文环境里可能有不同的内涵，它可以指一种通信模式、一种逻辑网络模型、一种技术、甚至一种理念。在P2P网络中如图1所示，所有通信节点的地位都是对等的，每个节点都扮演着客户机和服务器双重角色，节点之间通过直接通信实现文件信息、处理器运算能力、存储空间等资源的共享。P2P网络具有分散性、可扩展性、健壮性等特点，这使得P2P技术在信息共享、实时通信、协同工作、分布式计算、网络存储等领域都有广阔的应用。

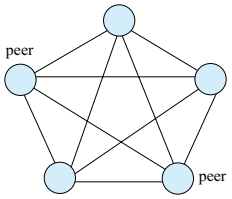


图1 P2P结构模型

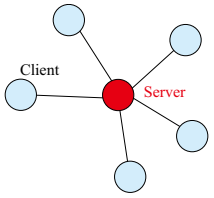


图2 CS模式

NAT简介

目前，IPv4地址资源的紧缺使得NAT技术获得了广泛的应用。NAT技术是一种把内部网络（简称为内网）私有IP地址转换为外部网络（简称为外网）公共IP地址的技术，它使得一定范围内的多台主机只利用一个公共IP地址连接到外网，可以在很大程度上缓解了公网IP地址紧缺的问题。

NAT对P2P通信的影响

NAT技术虽然在一定程度上解决了IPv4地址短缺的问题，在构建防火墙、保证网络安全方面都发挥了一定的作用，却破坏了端到端的网络通信。NAT阻碍主机进行P2P通信的主要原因是NAT不允许外网主机主动访问内网主机，因为NAT设备上没有相关转发表项，要在NAT网络环境中进行有效的P2P通信，就必须寻找相应的解决方案。本文就着重介绍几种常见的解决方案。

P2P穿越NAT的几种方案

反向链接技术

当通信的双方中只有一方位于NAT之后时，它们可以利用反向链接技术来进行P2P通信。图3中Client A（拥有内网IP地址10.0.0.1）位于NAT之后，它通过TCP端口1234连接到服务器（拥有外网IP地址）的TCP端口1235上，NAT设备（拥有外网IP地址155.99.25.11）为这个连接重新分配了TCP端口62000。Client B（拥有外网IP地址138.76.29.7）也通过TCP端口1234连接到服务器端口1235上。Client A和Client B从服务器处获知的对方的外网地址二元组{IP地址:端口号}分别为{138.76.29.7:1234}和{155.99.25.11:62000}，它们在各自的本地端口上进行侦听。

由于Client B 拥有外网IP地址，所以Client A要发起与Client B的通信，那么它可以直接通过TCP连接到Client B。但如果Client B尝试通过TCP连接到Client A进行P2P通信，则会失败，原因是Client A位于NAT设备后，虽然Client B发出的TCP SYN请求能够到达NAT设备的端口62000，但NAT设备会拒绝这个连接请求。要想与Client A通信，Client B要通过服务器给Client A转发一个连接请求，反过来请求Client A连接到Client B（即进行反向链接），从而建立起它们之间的TCP连接。

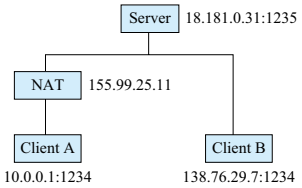


图3 反向链接示意图

UDP打洞技术

如果两个P2P客户端都位于NAT设备后面，想要进行P2P通信，那又该如何解决呢？UDP打洞技术就是为解决这个问题而应运而生的，它能够通过中间服务器实现P2P客户端互连。该技术在RFC 3027的第5.1节中有所提及，目前在多种在线游戏协议中已经得到了应用，下面来重点介绍下。

集中服务器

打洞技术假定客户端A和客户端B都可以与公网内的已知集中服务器建立UDP连接，一个客户端在集中服务器上登陆的时候，服务器记录下该客户端的两对地址二元组信息{IP地址:UDP端口}，一对是该客户端与集中服务器进行通信的自身的IP地址和端口号，另一对是集中服务器记录下的由服务器“观察”到的该客户端实际与自己通信所使用的IP地址和端口号。我们可以把前一对地址二元组看作是客户端的内网IP地址和端口号，把后一对地址二元组看作是客户端的内网IP地址和端口号经过NAT转换后的外网IP地址和端口号。集中服务器可以从客户端的登陆消息中得到该客户端的内网相关信息，还可以通过登陆消息的IP头和UDP头得到该客户端的外网相关信息。如果该客户端不是位于NAT设备后面，那么采用上述方法得到的两对地址二元组信息是完全相同的。

建立P2P的session

假定客户端A要发起对客户端B的直接连接，具体的“打洞”过程如下：

■ 客户端A最初不知道如何向客户端B发起连接，于是客户端A向集中服务器发送消息，请求集中服务器帮助建立与客户端B的UDP连接；

■ 集中服务器将含有客户端B的外网和内网的地址二元组发给客户端A，同时，集中服务器将包含有客户端A的外网和内网的地址二元组信息的消息也发给客户端B。这样一来，客户端A与客户端B就都知道对方外网和内网的地址二元组信息了；

■ 当客户端A收到由集中服务器发来的包含客户端B的外网和内网的地址二元组信息后，客户端A开始向客户端B的地址二元组发送UDP数据包，并且客户端A会自动锁定第一个给出响应的客户端B的地址二元组。同理，当客户端B收到由集中服务器发来的客户端A的外网和内网地址二元组信息后，也会开始向客户端A的外网和内网的地址二元组发送UDP数据包，并且自动锁定第一个得到客户端A回应的地址二元组。由于客户端A与客户端B互相向对方发送UDP数据包的操作是异步的，所以客户端A和客户端B发送数据包的时间先后并没有时序要求。

下面来看下这三者之间是如何进行UDP打洞的。在这我们分三种具体情景来讨论：

■ 第一种是最简单的一种情景，两个客户端都位于同一个NAT设备后面，即位于同一内网中；

■ 第二种是最普遍的一种情景，两个客户端分别位于不同的NAT设备后面，分属不同的内网；

■ 第三种是客户端位于两层NAT设备之后，通常最上层的NAT是由网络提供商提供的，第二层NAT是家用的NAT路由器之类的设备提供的。

P2P的两个客户端位于同一个NAT设备后面

首先假设两个客户端位于同一个NAT设备后面，并且位于内网，如图4所示。客户端A与集中服务器建立了UDP连接，经过NAT转换后，A的公网端口被映射为62000。客户端B同样与集中服务器建立了UDP连接，公网端口映射为62005。

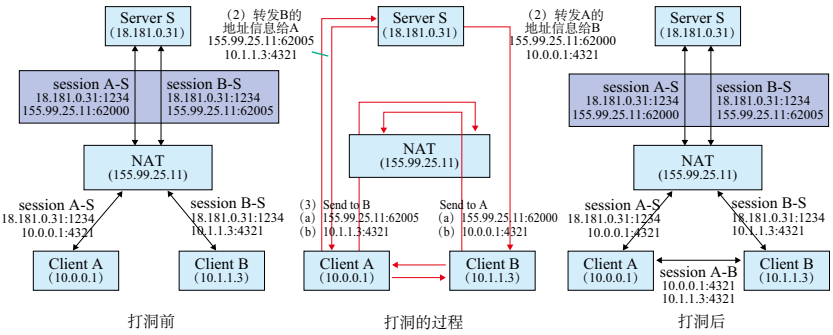


图4 位于同一个NAT设备后的UDP打洞过程

假设客户端A想通过集中服务器，发起对客户端B的连接。客户端A向集中服务器发出消息请求与客户端B进行连接，集中服务器将客户端B的外网地址二元组以及内网地址二元组发给客户端A，同时把客户端A的外网以及内网的地址二元组信息发给客户端B。客户端A和客户端B发往对方公网地址二元组信息的UDP数据包不一定会被对方收到，这取决于当前的NAT设备是否支持不同端口之间的UDP数据包能否到达即Hairpin转换特性，无论如何客户端A与客户端B发往对方内网的地址二元组信息的UDP数据包是一定可以到达的，内网数据包不需要路由，且速度更快。客户端A与客户端B推荐采用内网的地址二元组信息进行常规的P2P通信。

假定NAT设备支持Hairpin转换，具体的Hairpin转换见下一章节“P2P客户端位于多层NAT设备后面”，应用程序也应忽略与内网地址二元组的连接，如果客户端A、客户端B采用外网的地址二元组做为P2P通信的连接，这势必会造成数据包无谓地经过NAT设备，这是一种对资源的浪费。就目前的网络情况而言，应用程序在“打洞”的时候，最好还是把外网和内网的地址二元组都尝试一下。如果都能成功，优先以内网地址进行连接。

P2P客户端位于不同的NAT设备后面

假定客户端A与客户端B在不同的NAT设备后面，分属不同的内网，如图5所示。客户端A与客户端B都经由各自的NAT设备与集中服务器建立了UDP连接，客户端A与客户端B的本地端口号均为4321，集中服务器的公网端口号为1234。在向外的会话中，客户端A的外网IP被映射为155.99.25.11，外网端口为62000，客户端B的外网IP被映射为138.76.29.7，外网端口为31000。

如下所示：

客户端A——>本地IP:10.0.0.1，本地端口:4321，外网IP:155.99.25.11，外网端口:62000

客户端B——>本地IP:10.1.1.3，本地端口:4321，外网IP:138.76.29.7，外网端口:31000

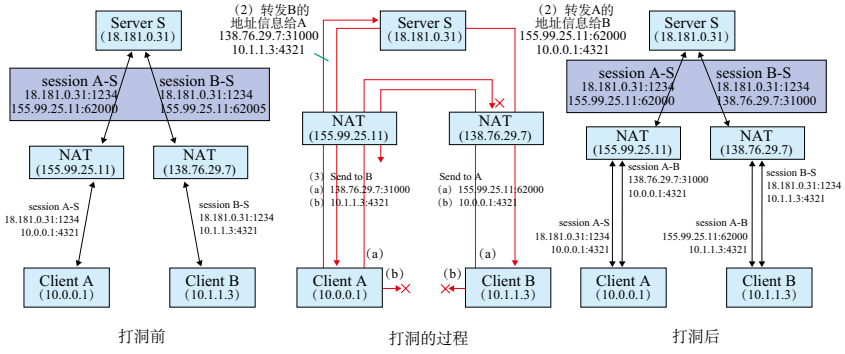


图5 位于不同NAT设备后的UDP打洞过程

在客户端A向服务器发送的登陆消息中，包含有客户端A的内网地址二元组信息，即10.0.0.1:4321；服务器会记录下客户端A的内网地址二元组信息，同时会把自己观察到的客户端A的外网地址二元组信息记录下来，即155.99.25.11:62000。同理，服务器也会记录下客户端B的内网地址二元组信息为10.1.1.3:4321和由服务器观察到的客户端B的外网地址二元组信息，138.76.29.7:31000。无论A与B二者中的任何一方向服务器发送P2P连接请求，服务器都会将其记录下来的上述的外网和内网地址二元组发送给A或B。

A、B分属不同的内网，它们彼此的内网地址在外网中是没有路由的，所以发往各自内网地址的UDP数据包会发送到错误的主机或者根本不存在的主机上。现在假定A的第一个消息将发往B的外网地址，如图5所示。该消息途经A的NAT设备，并在该设备上生成一个会话表项，该会话的源地址二元组信息是{10.0.0.1:4321}，该地址二元组信息和客户端A与服务器建立连接的时候NAT生成的源地址二元组信息一样，但它的目的地址不同。如果A的NAT设备给出的响应是OK的，那么A的NAT设备将保留A的内网地址二元组信息，并且所有来自A的源地址二元组信息为{10.0.0.1:4321}的数据包都沿用A与集中服务器事先建立起来的会话，外网地址二元组信息均为{155.99.25.11:62000}。

A向B的外网地址发送消息的过程就是“打洞”的过程，从A的内网的角度来看应为从{10.0.0.1:4321}发往{138.76.29.7:31000}，从A在其NAT设备上建立的会话来看，是从{155.99.25.11:62000}发到{138.76.29.7:31000}。

如果A发给B的外网地址二元组的消息包在B向A发送消息包之前到达B的NAT设备，B的NAT设备会认为A发过来的消息是未经授权的外网消息，会丢弃掉该数据包。B发往A的消息包与上述的过程一样，会在B的NAT设备上建立一个{10.1.1.3:4321，155.99.25.11:62000}的会话（通常也会沿用B与集中服务器连接时建立的会话，只是该会话现在不仅接受由服务器发给B的消息，还可以接受从A的NAT设备155.99.25.11:6200发来的消息），一旦A与B都向对方的NAT设备在外网上的地址二元组发送了数据包，就打开了A与B之间的“洞”，A与B向对方的外网地址发送数据，等效为向对方的客户端直接发送UDP数据包了。一旦应用程序确认已经可以通过往对方的外网地址发送数据包的方式让数据包到达NAT后面的目的应用程序，程序会自动停止继续发送用于“打洞”的数据包，转而开始真正的P2P数据传输。

P2P客户端位于多层NAT设备后面

有的网络拓扑结构包含了多个NAT设备，如果没有掌握该拓扑结构的详细信息，两个客户端之间是无法建立“最优化”的P2P路由的。现在我们来讨论最后一种情况，如图6所示。假定NAT C是由ISP（Internet Service Provider）提供的NAT设备，NAT C提供将多个用户节点映射到有限的几个公网IP的服务，NAT A和NAT B作为NAT C的内网节点将把用户的家庭网络或内部网络接入NAT C的内网，然后用户的内部网络就可以经由NAT C访问公网了。从这种拓扑结构上来看，只有服务器与NAT C是真正拥有公网可路由IP地址的设备，而NAT A和NAT B所使用的公网IP地址，实际上是由ISP服务提供商设定的（相对于NAT C而言）内网地址（本文的后续部分把这个由ISP提供的内网地址相对于NAT C称之为“伪”公网地址），同理隶属于NAT A与NAT B

的客户端，相对与NAT A、NAT B而言，它们处于NAT A、NAT B的内网，以此类推，客户端可以放到多层NAT设备后面。客户端A和客户端B发起对服务器S的连接的时候，就会依次在NAT A和NAT B上建立向外的session，而NAT A、NAT B要联入公网的时候，会在NAT C上再建立向外的session。

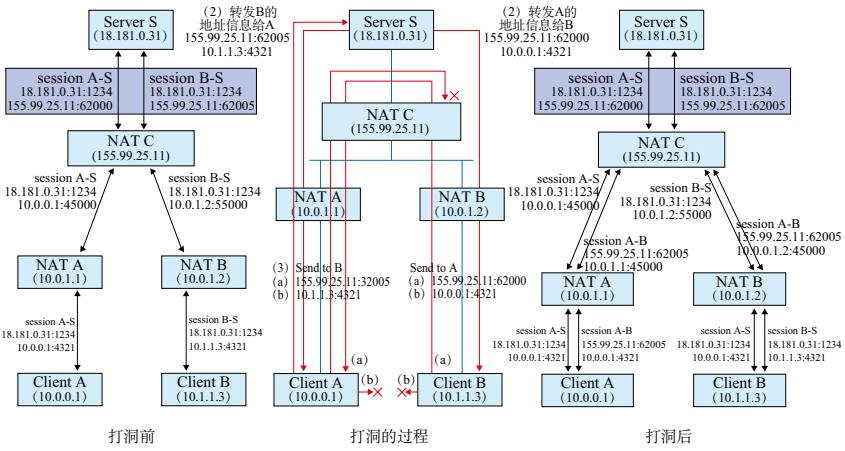


图6 多层NAT下的打洞过程

现在假定客户端A和B希望通过UDP“打洞”完成两个客户端的P2P直连。最优化的路由策略是客户端A向客户端B的“伪公网”IP上发送数据包，即ISP服务提供商指定的内网IP，NAT B的“伪”公网地址二元组，{10.0.1.2:55000}。由于从服务器的角度只能观察到真正的公网地址，也就是NAT A、NAT B在NAT C建立session的真正的公网地址{155.99.25.11:62000}以及{155.99.25.11:62005}，非常不幸的是客户端A与客户端B是无法通过服务器知道这些“伪”公网的地址，而且即使客户端A和B通过某种手段可以得到NAT A和NAT B的“伪”公网地址，我们仍然不建议采用上述的“最优化”的打洞方式，这是因为这些地址是由ISP服务提供商提供的或许会存在与客户端本身所在的内网地址重复的可能性（例如:NAT A的内网的IP地址域恰好与NAT A在NAT C的“伪”公网IP地址域重复，这样就会导致打洞数据包无法发出的问题）。

因此客户端别无选择，只能使用由公网服务器观察到的A，B的公网地址二元组进行“打洞”操作，用于“打洞”的数据包将由NAT C进行转发。

当客户端A向客户端B的公网地址二元组{155.99.25.11:62005}发送UDP数据包的时候，NAT A首先把数据包的源地址二元组由A的内网地址二元组{10.0.0.1:4321}转换为“伪”公网地址二元组{10.0.1.1:45000}，现在数据包到了NAT C，NAT C应该可以识别出来该数据包是要发往自身转换过的公网地址二元组，如果NAT C可以给出“合理”响应的话，NAT C将把该数据包的源地址二元组改为{155.99.25.11:62000}，目的地址二元组改为{10.0.1.2:55000}，即NAT B的“伪”公网地址二元组，NAT B最后会将收到的数据包发往客户端B。同样，由B发往A的数据包也会经过类似的过程。目前也有很多NAT设备不支持类似这样的“Hairpin转换”，但是已经有越来越多的NAT设备商开始加入对该转换的支持中来。

UDP在空闲状态下的超时问题

由于UDP转换协议提供的“洞”不是绝对可靠的，多数NAT设备内部都有一个UDP转换的空闲状态计时器，如果在一段时间内没有UDP数据通信，NAT设备会关掉由“打洞”操作打出来的“洞”，作为应用程序来讲如果想要做到与设备无关，就最好在穿越NAT以后设定一个穿越的有效期。

很遗憾目前没有标准有效期，这个有效期与NAT设备内部的配置有关，某些设备上最短的只有20秒左右。在这个有效期内，即使没有P2P数据包需要传输，应用程序为了维持该“洞”可以正常工作，也必须向对方发送“打洞”心跳包。这个心跳包是需要双方应用程序都发送的，只有一方发送不会维持另一方的session正常工作。除了频繁发送“打洞”心跳包以外，还有一个方法就是在当前的“洞”超时之前，P2P客户端双方重新“打洞”，丢弃原有的“洞”，这也不失为一个有效的方法。

关于TCP打洞技术

建立穿越NAT设备的P2P的TCP连接只比UDP复杂一点点，TCP协议的“打洞”从协议层来看是与UDP的“打洞”过程非常相似的。尽管

如此，基于TCP协议的打洞至今为止还没有被很好的理解，这也造成了对其提供支持的NAT设备不是很多。在NAT设备支持的前提下，基于TCP的“打洞”技术实际上与基于UDP的“打洞”技术一样快捷、可靠。实际上，只要NAT设备支持的话，基于TCP的P2P技术的健壮性将比基于UDP技术的更强一些，因为TCP协议的状态机给出了一种标准的方法来精确的获取某个TCP session的生命期，而UDP协议则无法做到这一点。

套接字和TCP端口的重用

实现基于TCP协议的P2P打洞过程中，最主要的问题不是来自于TCP协议，而是来自于应用程序的API接口。这是由于标准的伯克利（Berkeley）套接字的API是围绕着构建客户端/服务器程序而设计的，API允许TCP流套接字通过调用connect（）函数来建立向外的连接，或者通过listen（）和accept函数接受来自外部的连接，但是，API不提供类似UDP那样的，同一个端口既可以向外连接，又能够接受来自外部的连接。而且更糟的是，TCP的套接字通常仅允许建立1对1的响应，即应用程序在将一个套接字绑定到本地的一个端口以后，任何试图将第二个套接字绑定到该端口的操作都会失败。

为了让TCP“打洞”能够顺利工作，我们需要使用一个本地的TCP端口来监听来自外部的TCP连接，同时建立多个向外的TCP连接。幸运的是，所有的主流操作系统都能够支持特殊的TCP套接字参数，通常叫做“SO_REUSEADDR”，该参数允许应用程序将多个套接字绑定到本地的一个地址二元组（只要所有要绑定的套接字都设置了SO_REUSEADDR参数即可）。BSD系统引入了SO_REUSEPORT参数，该参数用于区分端口重用还是地址重用，在这样的系统里面，上述所有的参数必须都设置才行。

打开P2P的TCP流

假定客户端A希望建立与B的TCP连接。我们像通常一样假定A和B已经与公网上的已知服务器建立了TCP连接。服务器记录下来每个接入的客户端的公网和内网的地址二元组，如同为UDP服务的时候一样。从协议层来看，TCP“打洞”与UDP“打洞”是几乎完全相同的过程。

- 客户端A使用其与服务器的连接向服务器发送请求，要求服务器协助其连接客户端B；
- 服务器将B的公网和内网的TCP地址的二元组信息返回给A，同时，服务器将A的公网和内网的地址二元组也发送给B；
- 客户端A和B使用连接服务器的端口异步地发起向对方的公网、内网地址二元组的TCP连接，同时监听各自的本地TCP端口是否有外部的连接联入；
- A和B开始等待向外的连接是否成功，检查是否有新连接联入。如果向外的连接由于某种网络错误而失败，如：“连接被重置”或者“节点无法访问”，客户端只需要延迟一小段时间（例如延迟一秒钟），然后重新发起连接即可，延迟的时间和重复连接的次数可以由应用程序编写者来确定；
- TCP连接建立起来以后，客户端之间应该开始鉴权操作，确保目前联入的连接就是所希望的连接。如果鉴权失败，客户端将关闭连接，并且继续等待新的连接联入。客户端通常采用“先入为主”的策略，只接受第一个通过鉴权操作的客户端，然后将进入P2P通信过程不再继续等待是否有新的连接联入。

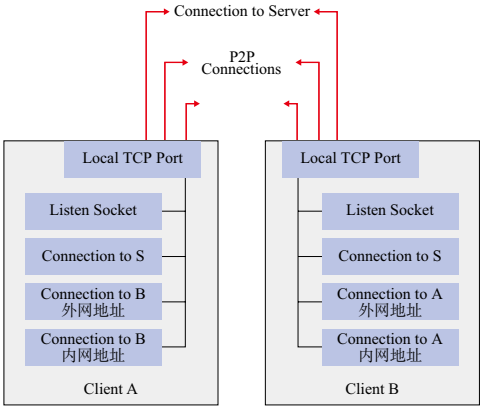


图7 TCP打洞

与UDP不同的是，因为使用UDP协议的每个客户端只需要一个套接字即可完成与服务器的通信，而TCP客户端必须处理多个套接字绑定到同一个本地TCP端口的问题，如图7所示。现在来看实际中常见的一种情景，A与B分别位于不同的NAT设备后面，如图5所示，并且假定图中的端口号是TCP协议的端口号，而不是UDP的端口号。图中向外的连接代表A和B向对方的内网地址二元组发起的连接，这些连接或许会失败或者无法连接到对方。如同使用

UDP协议进行“打洞”操作遇到的问题一样，TCP的“打洞”操作也会遇到内网的IP与“伪”公网IP重复造成连接失败或者错误连接之类的问题。

客户端向彼此公网地址二元组发起连接的操作，会使得各自的NAT设备打开新的“洞”允许A与B的TCP数据通过。如果NAT设备支持TCP“打洞”操作的话，一个在客户端之间的基于TCP协议的流通道就会自动建立起来。如果A向B发送的第一个SYN包发到了B的NAT设备，而B在此前没有向A发送SYN包，B的NAT设备会丢弃这个包，这会引起A的“连接失败”或“无法连接”问题。而此时，由于A已经向B发送过SYN包，B发往A的SYN包将被看作是由A发往B的包的回应的一部分，所以B发往A的SYN包会顺利地通过A的NAT设备，到达A，从而建立起A与B的P2P连接。

从应用程序的角度来看TCP“打洞”

从应用程序的角度来看，在进行TCP“打洞”的时候都发生了什么呢？假定A首先向B发出SYN包，该包发往B的公网地址二元组，并且被B的NAT设备丢弃，但是B发往A的公网地址二元组的SYN包则通过A的NAT到达了A，然后，会发生以下的两种结果中的一种，具体是哪一种取决于操作系统对TCP协议的实现：

■ A的TCP实现会发现收到的SYN包就是其发起连接并希望联入的B的SYN包，通俗一点来说就是“说曹操，曹操到”的意思，本来A要去找B，结果B自己找上门来了。A的TCP协议栈因此会把B作为A向B发起连接connect的一部分，并认为连接已经成功。程序A调用的异步connect（）函数将成功返回，A的listen（）等待从外部联入的函数将没有任何反映。此时，B联入A的操作在A程序的内部被理解为A联入B连接成功，并且A开始使用这个连接与B开始P2P通信。

由于收到的SYN包中不包含A需要的ACK数据，因此，A的TCP将用SYN-ACK包回应B的公网地址二元组，并且将使用先前A发向B的SYN包一样的序列号。一旦B的TCP收到由A发来的SYN-ACK包，则把自己的ACK包发给A，然后两端建立起TCP连接。简单的说，第一种，就是即使A发往B的SYN包被B的NAT丢弃了，但是由于B发往A的包到达了A。结果是，A认为自己连接成功了，B也认为自己连接成功了，不管是谁成功了，总之连接是已经建立起来了。

■ 另外一种结果是，A的TCP实现没有像前文中所讲的那么“智能”，它没有发现现在联入的B就是自己希望联入的。就好比在机场接人，明明遇到了自己想要接的人却不认识，误认为是其他的人，安排别人给接走了，后来才知道是自己错过了机会，但是无论如何，人已经接到了任务已经完成了。然后，A通过常规的listen（）函数和accept（）函数得到与B的连接，而由A发起的向B的公网地址二元组的连接会以失败告终。尽管A向B的连接失败，A仍然得到了B发起的向A的连接，等效于A与B之间已经联通，不管中间过程如何，A与B已经连接起来了，结果是A和B的基于TCP协议的P2P连接已经建立起来了。

第一种结果适用于基于BSD的操作系统对于TCP的实现，而第二种结果更加普遍一些，多数Linux和Windows系统都会按照第二种结果来处理。

SR8800路由器NAT处理流程

文/王军

NAT功能众所周知，是为了解决IPv4地址不足而产生的地址转换技术，主要包括静态NAT、动态NAT和NAT Server技术。从NAT转换发生的位置来说，NAT又有入方向和出方向NAT之分。而随着MPLS VPN技术的应用，在现实组网中也存在VPN内部NAT、不同VPN之间需要通过NAT互访等需求，VPN NAT技术也随之出现。本文介绍了SR8800设备NAT处理流程，并对设备NAT转换的各个过程进行详细介绍，也指出了设备自身NAT功能的一些特点。

SR8800设备NAT处理结构分析

各个设备对于NAT功能的实现因其设备自身软硬件情况不同而不同，对于SR8800系列路由器设备，第一代业务板卡不支持NAT特性，需要单独的NAT板卡才能实现NAT转换，第二代的业务板卡支持NAT特性，在业务板卡上增加多核CPU的方式来处理NAT转换，但其NAT转换性能远不及单独NAT板卡处理，仍然建议使用业务板卡＋NAT板卡的方式进行配置。

NAT转换需要高性能处理器的支撑，盒式设备一般采用CPU处理NAT转换，当NAT会话新建、并发数量很大时，会占用大量的CPU资源，对设备CPU的性能要求很高。框式设备的NAT转换可以采用分布式、集中式，分布式NAT是在业务板卡上，增加处理器的方式，每块业务板卡单独处理自己的NAT转换，这样一来，会增加每块板卡的成本，且每块板卡的处理性能不高，对于大容量的NAT转换业务，很难集中到一块业务板卡上，还会给网络规划带来难度；SR8800的NAT板卡使用NP高性能处理器集中处理NAT转换，不仅单块NAT板卡的性能远远高于分布式NAT，并且支持多块业务板卡负载分担方式来提高整机NAT转换性能。

SR8800设备NAT处理流程

SR8800 NAT板卡上无业务口，数据流量均由业务板卡转发过来，从而在NAT板卡上进行对相应数据流量的源目的IP转换工作。NAT转换后的流量再转发给相应的出口业务板卡进行转发。总体而言，设备NAT处理过程主要包括三部分：流量标记过程、NAT转换过程、NAT流量转发过程，具体流程图见图1。

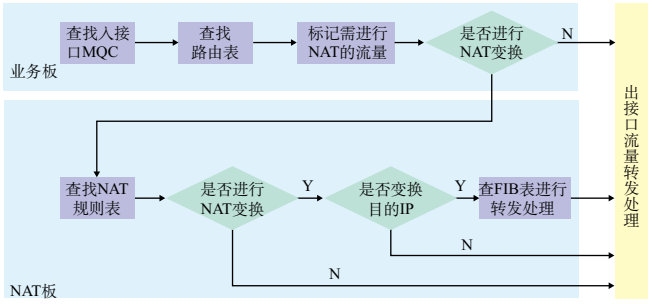


图1 SR8800设备NAT处理流程

流量标记过程

流量标记由业务板卡完成，主要是根据设备上的配置对需要进行NAT的流量进行“NAT标记”，然后将所有带有NAT标记的流量均转发至NAT板卡，而对于没有NAT标记的流量则直接正常转发。对于哪些流量会被打上NAT标记，设备依据如下原则进行判断：

- 入接口配置了MQC将流量重定向至NAT板，对于满足此类别的流量进行NAT标记；
- 在配置NAT后，设备会自动生成去往NAT地址池IP的32位路由，对于匹配此目的IP的流量进行NAT标记；
- 出接口配置了NAT outbound命令且在NAT板卡视图下绑定了此接口时，对于所有从此接口转发的流量进行NAT标记。

VPN内始发的私网数据和远端PE设备发送到本设备私网的MPLS报文，其流量标记判断原则相同，只是对于远端PE发送过来的MPLS报文，需要在弹出标签再进行标记判断处理。

NAT转换过程

流量被转发至NAT板卡上后，板卡根据设备配置所生成的NAT规则对数据流进行转换此处NAT规则的设定决定了设备进行的NAT功能。SR8800设备较H3C其他路由器存在着其自身NAT功能的特点：

■ 能够实现入方向静态/动态NAT

由于利用NAT单板单独对NAT进行处理，因而在H3C路由器设备普遍的出方向NAT基础之上，SR8800能够通过在入方向配置MQC方式进行入方向NAT。

■ 能够基于不同目的地址、出接口或下一跳实现对相同源地址的不同NAT映射

在NAT规则上，SR8800设备对映射表项进行了细化，转换映射更加灵活。对于静态NAT映射，在实现了全局静态映射的基础上，SR8800还可以根据目的网段进行不同的源地址映射，也可以实现多出口时不同源地址的转换，而对于同一个以太网出口，还可以精确地根据不同的下一跳地址进行映射。动态NAT也支持将映射关系与不同的出接口及下一跳地址进行关联。

■ 能够手工配置静态NAT的方向性

通过在配置静态NAT映射时增加unidirectional参数，可以控制数据访问必须由内网主机向外网始发。而不配置unidirectional参数时，数据访问可以由内网或外网主机始发。

当设备上配置的NAT规则较多时，设备按照如下优先级对报文进行转换：

■ 当存在静态转换和动态转换时，静态优先；

■ 都是动态转换时，根据ACL中源IP的反掩码决定，精确的优先；

■ 如果动态规则的反掩码程度相同，则配置了下一跳和出接口规则的优先。

NAT流量转发过程

对于源地址变换，业务板卡在将做了NAT标记的流量转发至NAT板卡上之前会查询FIB表并将转发信息上送到NAT板卡，NAT板卡在完成NAT转换过程后，依据此转发信息将NAT后的流量直接转发至出接口；

对于目的地址变换，NAT板卡在完成NAT转换过程后，直接在NAT板卡上查找FIB转发表以将NAT后的流量转发至出接口；

NAT后的流量被转发到出接口上后，会按照设备的出接口转发处理流程进行处理。

总结

本文就SR8800设备的NAT处理流程进行了说明，指出了设备NAT功能的一些特点，在现实组网中，将会有各种复杂的NAT组合需求出现，通过对NAT技术的理解、对设备具体NAT处理流程的掌握，以及对设备NAT特点的熟知，可以灵活应用以满足实际需求。

NAT ALG原理与应用

文/曹佐清

NAT ALG简介

普通NAT实现了对UDP或TCP报文头中的IP地址及端口转换功能，但对应用层数据载荷中的字段无能为力，在许多应用层协议中，比如多媒体协议（H.323、SIP等）、FTP、SQLNET等，TCP/UDP载荷中带有地址或者端口信息，这些内容不能被NAT进行有效的转换，就可能导致问题。而NAT ALG（Application Level Gateway，应用层网关）技术能对多通道协议进行应用层报文信息的解析和地址转换，将载荷中需要进行地址转换的IP地址和端口或者需特殊处理的字段进行相应的转换和处理，从而保证应用层通信的正确性。

例如，FTP应用就由数据连接和控制连接共同完成，而且数据连接的建立动态地由控制连接中的载荷字段信息决定，这就需要ALG来完成载荷字段信息的转换，以保证后续数据连接的正确建立。

NAT ALG特点

NAT ALG为内部网络和外部网络之间的通信提供了基于应用的访问控制，具有以下优点：

■ ALG统一对各应用层协议报文进行解析处理，避免其它模块对同一类报文应用层协议的重复解析，可以有效提高报文转发效率；

■ 可支持多种应用层协议：FTP、H.323（包括RAS、H.225、H.245）、SIP、DNS、ILS、MSN/QQ、NBT、RTSP、SQLNET、TFTP等。

NAT ALG技术实现

先介绍ALG涉及到的两个概念：

会话：记录了传输层报文之间的交互信息，包括源IP地址、源端口、目的IP地址、目的端口、协议类型和源/目的IP地址所属的

VPN实例。交互信息相同的报文属于一条流，通常情况下，每个会话对应出方向和入方向的两条流。

动态通道：当应用层协议报文中携带地址信息时，这些地址信息会被用于建立动态通道，后续符合该地址信息的连接将使用已经建立的动态通道来传输数据。

下面以多通道应用协议FTP在NAT组网环境中的ALG应用来具体说明报文载荷的转换过程。

ALG与FTP的应用

FTP的两种不同工作模式：PORT（主动模式）与PASV（被动模式）。

FTP需要用到两个连接：控制连接与数据连接，控制连接专门用于FTP控制命令及命令执行信息传送；数据连接专门用于传输数据（上传/下载）。

主动模式（PORT）的连接过程

如图1所示，位于内部网络的客户端以PORT方式访问外部网络的FTP服务器，经过中间的设备进行NAT转换，该设备上使能了ALG特性。

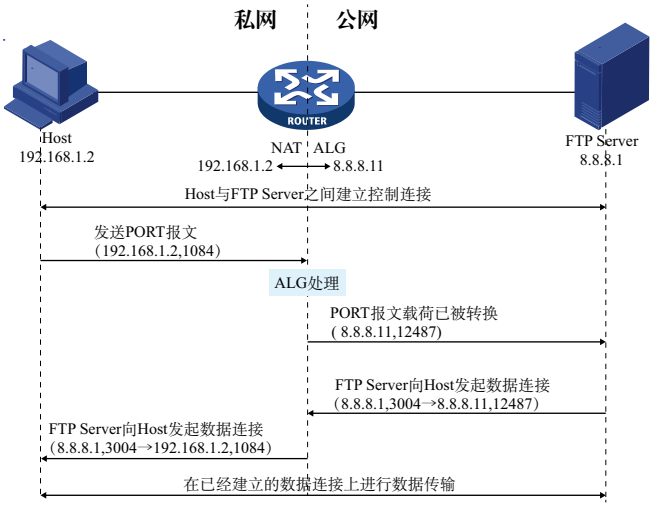


图1 FTP PORT方式报文载荷的ALG处理图

图1中私网侧的主机要访问公网的FTP服务器。NAT设备上配置了私网地址192.168.1.2到公网地址8.8.8.11的映射，实现地址的NAT转换，以支持私网主机对公网的访问。组网中，若没有ALG对报文载荷的处理，私网主机发送的PORT报文到达服务器端后，服务器无法根据私网地址进行寻址，也就无法建立正确的数据连接。整个通信过程包括如下四个阶段：

■ 私网主机和公网FTP服务器之间通过TCP三次握手成功建立控制连接；

■ 控制连接建立后，私网主机向FTP服务器发送PORT报文，报文中携带私网主机指定的数据连接的目的地址和端口，用于通知服务器使用该地址和端口和自己进行数据连接；

■ PORT报文在经过支持ALG特性的NAT设备时，报文载荷中的私网地址和端口会被转换成对应的公网地址和端口。即设备将收到的PORT报文载荷中的私网地址192.168.1.2转换成公网地址8.8.8.11，端口1084转换成12487；

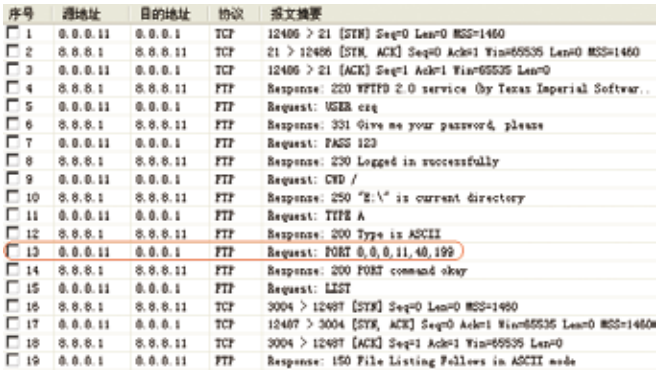
■ 公网的FTP服务器收到PORT报文后，解析其内容，并向私网主机发起数据连接，该数据连接的目的地址为8.8.8.11，目的端口为12487（注意：一般情况下，该报文源端口为20，但由于FTP协议没有严格规定，有的服务器发出的数据连接源端口为大于1024的随机端口，如本例采用的是wftpd服务器，采用的源端口为3004）。由于该目的地址是一个公网地址，因此后续的数据连接就能够成功建立，从而实现私网主机对公网服务器的访问。

在HOST（FTP客户端）抓包如图2所示：

序号	源地址	目的地址	协议	报文摘要
1	192.168.1.2	8.8.8.1	TCP	1083 > 21 [SYN] Seq=0 Len=0 MSS=1460
2	8.8.8.1	192.168.1.2	TCP	21 > 1083 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 M...
3	192.168.1.2	8.8.8.1	TCP	1083 > 21 [ACK] Seq=1 Ack=1 Win=65535 [TCP CHECKSU...
4	8.8.8.1	192.168.1.2	FTP	Response: 220 WFTPD 2.0 service (by Texas Imperial...
5	192.168.1.2	8.8.8.1	FTP	Request: USER cnq
6	8.8.8.1	192.168.1.2	FTP	Response: 331 Give me your password, please
7	192.168.1.2	8.8.8.1	FTP	Request: PASS 123
8	8.8.8.1	192.168.1.2	FTP	Response: 230 Logged in successfully
9	192.168.1.2	8.8.8.1	FTP	Request: CWD /
10	8.8.8.1	192.168.1.2	FTP	Response: 250 "E:\" is current directory
11	192.168.1.2	8.8.8.1	FTP	Request: TYPE A
12	8.8.8.1	192.168.1.2	FTP	Response: 200 Type is ASCII
13	192.168.1.2	8.8.8.1	FTP	Request: PORT 192,168,1,2,4,60
14	8.8.8.1	192.168.1.2	FTP	Response: 200 PORT command okay
15	192.168.1.2	8.8.8.1	FTP	Request: LIST
16	8.8.8.1	192.168.1.2	TCP	3004 > 1084 [SYN] Seq=0 Len=0 MSS=1460
17	192.168.1.2	8.8.8.1	TCP	1084 > 3004 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0...
18	8.8.8.1	192.168.1.2	TCP	3004 > 1084 [ACK] Seq=1 Ack=1 Win=65535 Len=0
19	8.8.8.1	192.168.1.2	FTP	Response: 150 File Listing Follows in ASCII mode

图2 PORT模式FTP客户端抓包

在FTP服务器端抓包如图3所示：



序号	源地址	目的地址	协议	报文摘要
1	8.8.8.11	8.8.8.1	TCP	12486 > 21 [SYN] Seq=0 Len=0 MSS=1460
2	8.8.8.1	8.8.8.11	TCP	21 > 12486 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
3	8.8.8.11	8.8.8.1	TCP	12486 > 21 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	8.8.8.1	8.8.8.11	FTP	Response: 220 WFTPD 2.0 service (by Texas Imperial Softwar..
5	8.8.8.11	8.8.8.1	FTP	Request: USER cxq
6	8.8.8.1	8.8.8.11	FTP	Response: 331 Give me your password, please
7	8.8.8.11	8.8.8.1	FTP	Request: PASS 123
8	8.8.8.1	8.8.8.11	FTP	Response: 230 Logged in successfully
9	8.8.8.11	8.8.8.1	FTP	Request: CWD /
10	8.8.8.1	8.8.8.11	FTP	Response: 250 "E:\" is current directory
11	8.8.8.11	8.8.8.1	FTP	Request: TYPE A
12	8.8.8.1	8.8.8.11	FTP	Response: 200 Type is ASCII
13	8.8.8.11	8.8.8.1	FTP	Request: PORT 8,8,8,11,48,199
14	8.8.8.1	8.8.8.11	FTP	Response: 200 PORT command okay
15	8.8.8.11	8.8.8.1	FTP	Request: LIST
16	8.8.8.1	8.8.8.11	TCP	3004 > 12487 [SYN] Seq=0 Len=0 MSS=1460
17	8.8.8.11	8.8.8.1	TCP	12487 > 3004 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
18	8.8.8.1	8.8.8.11	TCP	3004 > 12487 [ACK] Seq=1 Ack=1 Win=65535 Len=0
19	8.8.8.1	8.8.8.11	FTP	Response: 150 File Listing Follows in ASCII mode

图3 PORT模式FTP服务器端抓包

由上抓包可知：主动模式（PORT）的连接过程是：客户端程序首先会为自己随机分配一个TCP 端口，它使用这个端口向服务器的FTP端口（默认为21）发出连接请求，服务器接受请求之后会建立一条控制链路，然后客户程序向服务器发出PORT命令（通常格式为PORT A1，A2，A3，A4，P1，P2，其中A1，A2，A3，A4为客户端IP地址，P1，P2为随机的一个数据连接端口号，端口号等于P1*256＋P2），告诉服务器它的数据通道的端口打开了。当需要传送数据时，服务器向客户端提供的随机端口发送连接请求，请求被接受之后便开始传输数据，主动模式下，需要做ALG处理的是客户端发出的PORT报文，如FTP客户端上抓包中的第13个报文，其中有一个包含地址和端口的字段为Request arg，如图4所示：

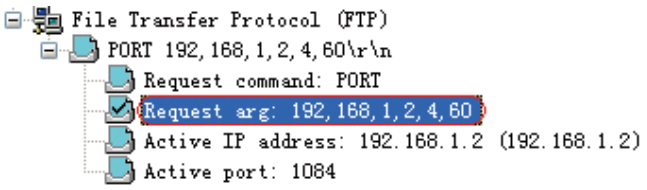


图4 ALG转换前的PORT报文

FTP客户端发出的PORT报文经过NAT设备后对应FTP服务器端上抓的第13个报文，私网地址192.168.1.2转换成公网地址8.8.8.11，端口1084转换成12487，如图5所示：

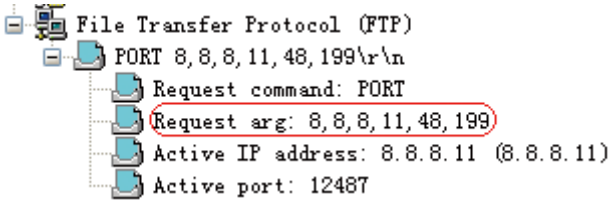


图5 ALG转换后的PORT报文

被动模式（PASV）的连接过程

如图6所示，位于外部网络的FTP客户端以PASV方式访问内部网络的FTP服务器，经过中间的设备进行NAT转换，该设备上使能了ALG特性。

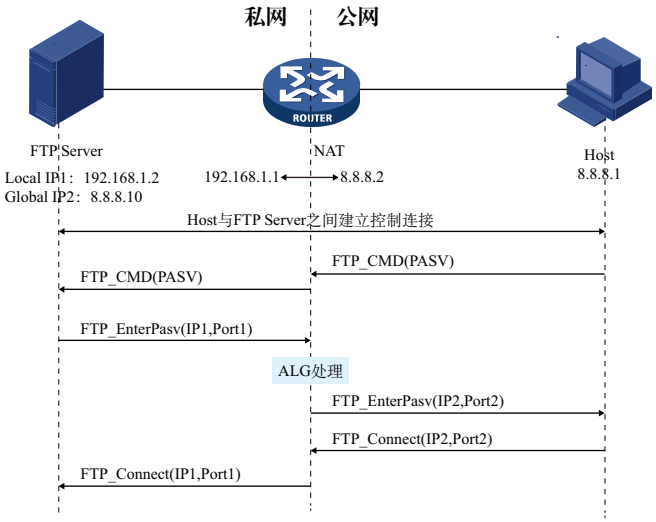


图6 FTP PASV方式报文载荷的ALG处理图

图6中的NAT设备上配置了私网地址192.168.1.1到公网地址8.8.8.2的映射，实现地址的NAT转换。整个通信过程包括如下四个阶段：

■ 建立控制通道

客户端向服务器发送TCP连接请求。TCP连接建立成功后，服务器和客户端进入用户认证阶段。若TCP连接失败，服务器会断开与客户端的连接。

■ 用户认证

客户端向服务器发送认证请求，报文中包含FTP命令（USER、PASSWORD）及命令所对应的内容。客户端发送的认证请求报

文在通过配置了ALG的设备时，报文载荷中携带的命令字将会被解析出来，用于进行状态机转换过程是否进行了正确的检查。若状态机转换发生错误，则丢弃报文。这样可防止客户端发送状态机错误的报文攻击服务器或者非法登录服务器，起到保护服务器的作用。客户端的认证请求报文通过ALG处理之后，到达服务器端，服务器将对其进行响应。

■ 创建数据通道

认证状态正确且用户是服务器已经授权的客户端，才能和服务器建立数据连接，进行数据的交互。如图6所示，当客户端发送“PASV”命令发起连接时，服务器会在发送给客户端的PASV响应报文中携带自己的私网地址和端口号（IP1，Port1），响应报文经过ALG设备时被解析，其中携带的服务器的私网地址和端口号被转换成其对应的公网地址和端口号（IP2，Port2），之后在该地址和端口与客户端的地址和端口之间建立起数据通道。

■ 数据交互

客户端和服务器之间的数据交互可以直接通过数据通道来进行。

在FTP服务器端抓包如图7所示：

序号	源地址	目的地址	协议	报文摘要
1	8.8.8.1	192.168.1.2	TCP	3318 > 21 [SYN] Seq=0 Len=0 MSS=1460
2	192.168.1.2	8.8.8.1	TCP	21 > 3318 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
3	8.8.8.1	192.168.1.2	TCP	3318 > 21 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	192.168.1.2	8.8.8.1	FTP	Response: 220 WFTPD 2.0 service (by Texas Imperial Soft...
5	8.8.8.1	192.168.1.2	FTP	Request: USER cxq
6	192.168.1.2	8.8.8.1	FTP	Response: 331 Give me your password, please
7	8.8.8.1	192.168.1.2	FTP	Request: PASS 123
8	192.168.1.2	8.8.8.1	FTP	Response: 230 Logged in successfully
9	8.8.8.1	192.168.1.2	FTP	Request: CWD /
10	192.168.1.2	8.8.8.1	FTP	Response: 250 "D:\" is current directory
11	8.8.8.1	192.168.1.2	FTP	Request: TYPE A
12	192.168.1.2	8.8.8.1	FTP	Response: 200 Type is ASCII
13	8.8.8.1	192.168.1.2	FTP	Request: PASV
14	192.168.1.2	8.8.8.1	FTP	Response: 227 Entering Passive Mode (192,168,1,2,4,162)
15	8.8.8.1	192.168.1.2	TCP	3319 > 1186 [SYN] Seq=0 Len=0 MSS=1460
16	192.168.1.2	8.8.8.1	TCP	1186 > 3319 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=...
17	8.8.8.1	192.168.1.2	TCP	3319 > 1186 [ACK] Seq=1 Ack=1 Win=65535 Len=0
18	8.8.8.1	192.168.1.2	FTP	Request: LIST
19	192.168.1.2	8.8.8.1	FTP	Response: 150 File Listing Follows in ASCII mode
20	192.168.1.2	8.8.8.1	FTP-DATA	FTP Data: 673 bytes

图7 PASV模式FTP服务器端抓包

由上抓包可知：被动模式（PASV）的连接过程是客户端程序首先为自己随机分配一个TCP端口，使用这个端口向服务器的FTP端口发出连接请求，服务器接受请求之后会建立一条控制链路，然后客户端程序发出PASV命令，要求服务器采用PASV模式建立数据连接，服务器便为自己随机分配一个数据通道端口，并将这个端口号告诉客户端程序（通常格式为：Entering Passive Mode（A1，A2，A3，A4，P1，P2），其中A1，A2，A3，A4为服务器IP地址，P1，P2为随机端口号）。当需要传送数据时，客户端程序采用另一个随机端口向服务器提供的数据通道端口发送连接请求，请求被接受之后便开始传输数据，数据链路通道打开，被动模式下，需要做ALG处理的是服务器发出的Pasv response报文，如上抓包中的第14个包，其中有一个包含地址和端口的字段为Response arg，如图8所示：

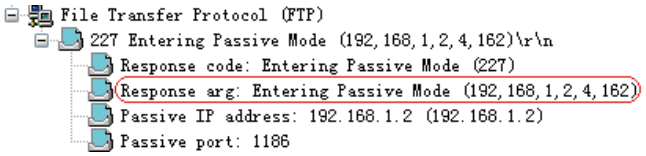


图8 需要做ALG的Pasv response报文

ALG与H.323/SIP的应用

ALG与H.323的应用

H.323协议用于发起会话，它能控制多个参与者参加的多媒体会话的建立和终结，并能动态调整和修改会话属性，如会话带宽要求、传输的媒体类型（语音、视频等）、媒体的编解码格式、广播的支持等。

H.323协议采用Client/Server模型，如在图9所示的语音组网中，主要通过网关（Gateway）与网守（Gatekeeper）之间的通信来完成用户呼叫的建立过程。

网关（Gateway）：用于连接H.323电话终端；

网守（Gatekeeper）：注册/位置/代理服务器，管理各Gateway

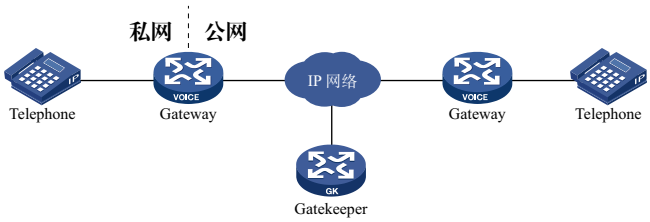


图9 H.323语音应用组网图

H.323协议栈如图10所示：

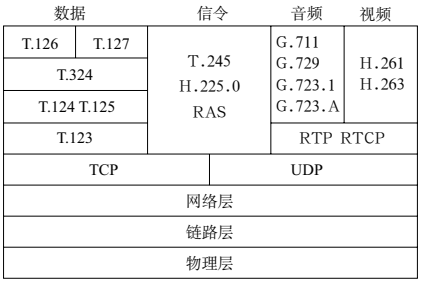


图10 H.323协议栈

由协议栈可知，H.323是一个协议族，由众多协议来完成地址定位、注册、媒体协商等一系列工作。其中TCP/UDP载荷中带有地址或者端口信息，若在网关进行了NAT处理后，则需要进行ALG处理的有H.225、H.245、RAS等信令协议报文，具体为：

■ UDP RAS报文：gatekeeper、registar、admission的request和confirm报文；

■ TCP H.225报文：setup、alerting、connect报文；

■ TCP H.245报文：open logical chanel、open logical chanel ACK报文。

H.323通信中会建立4种连接：

RAS连接：网关与网守之间的UDP连接，源目的端口均是1719。

H.225连接：routed模式建立在网关与网守之间，redirect模式建立在网关与网关之间的TCP连接。源端口一般是发起方随机分配的，目的端口是1720。

H.245连接：routed模式建立在网关与网守之间，redirect模式建立在网关与网关之间的TCP连接。源端口一般是发起方随机分配的，目的端口通过H.225的connect报文协商。

RTP/RTCP连接：建立在网关与网关之间的UDP连接。源和目的通过H.245连接里的open logical chanel及其ACK报文协商。

ALG与SIP的应用

SIP（session Initiation Protocol，会话初始协议）是一个用于建立、更改和终止多媒体会话的应用层控制协议，其中的会话可以是IP电话、多媒体会话或多媒体会议。SIP是通过各种头域里的信息的交互来管理会话的。而头域里与呼叫建立相关的包含IP地址和端口信息的字段需要被ALG处理，否则无法正确进行呼叫。SIP UA直接呼叫组网如图11所示（两台路语音由器作为SIP UA，能够互相直接呼叫）：

图11 SIP UA直接呼叫组网图

SIP消息采用文本方式编码，包括请求消息与响应消息两类。SIP请求消息包括如下六种。

■ INVITE：用于邀请用户加入一个呼叫。

■ ACK：用于对请求消息的响应消息进行确认。

■ OPTIONS：用于请求协商能力信息。

■ BYE：用于释放已建立的呼叫。

■ CANCEL：用于释放尚未建立的呼叫。

■ REGISTER：用于向SIP注册服务器登记用户位置等信息。

SIP响应消息用于对请求消息进行响应，指示呼叫或注册的成功或失败状态。在请求与响应报文中需要进行ALG处理的地址字段类型主要有：Via、Record_Route、Contact、SDP。

ALG处理流程为如下三个步骤：

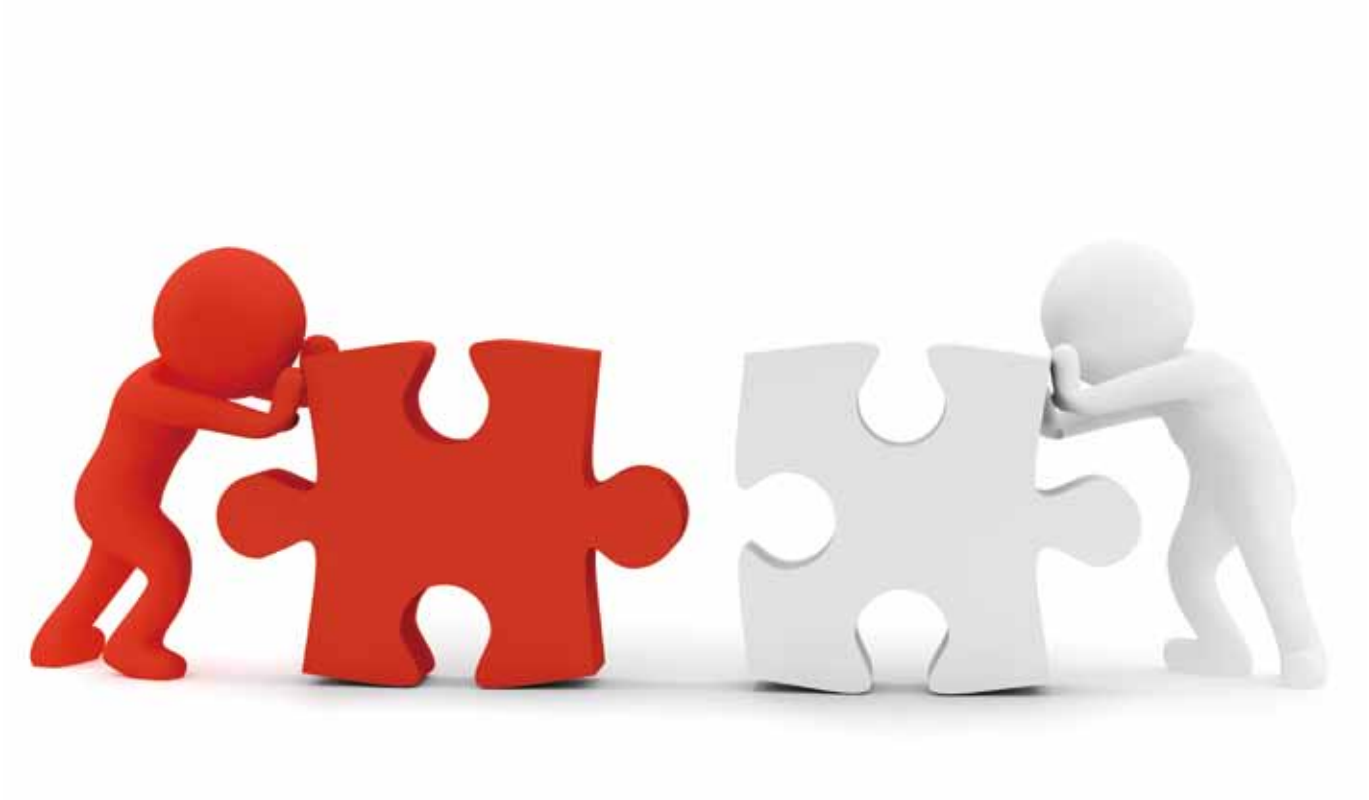
首先，ALG根据会话标识的协议类型对报文进行解码，若解码发现报文为不需要做ALG或解码发现为错误字段时退出，解码发现需进行字段转换时进一步处理；

其次，ALG查找接口上的NAT配置，根据NAT配置转换报文中的IP地址、端口、call-id等信息并建立关联表，关联表记录了载荷地址的转换关系；

最后，ALG调整报文载荷中的长度字段，如sip message header的content-length字段标识message body的长度，ALG对message body中的地址转换后，message body长度可能变化，content-length字段值需要置为变化后的值。

传统VPN与NAT穿越的兼容性

文/任俊峰



概述

VPN（Virtual Private Network，虚拟专用网）是一种基于公共数据网的服务，它依靠ISP（Internet Service Provider）和NSP（Network Service Provider），在公共网络中建立虚拟专用通信网络。VPN可以极大地降低用户的费用，并且提供比传统专线方式更强的安全性。

在VPN中广泛使用了各种各样的隧道技术，有二层隧道技术，也有三层隧道技术。常用隧道协议包括：L2TP、GRE、IPsec、SSL VPN等。那么，什么是隧道呢?

隧道是一种封装技术，它利用一种网络协议来传输另一种网络协议，即利用一种网络传输协议，将其他协议产生的数据报文封装在它自己的报文中，然后在网络中传输。实际上隧道可以看作一个虚拟的点到点连接。

隧道技术简单地说就是：原始报文在A地进行封装，到达B地后把封装去掉，还原成原始报文，这样就形成了一条由A到B的通信隧道。隧道技术就是指包括数据封装、传输和解封装在内的全过程。

NAT（本文如无特殊说明，NAT包括端口转换PAT情况）主要用于解决IPv4地址紧缺问题，在目前网络中NAT应用非常广泛，特

别是在企业网出口网关大都使用了NAT技术解决公网地址不足的问题。

如图1所示，当企业分支机构处于一个园区或者小运营商范围内，企业分支出口的IP地址很可能不是公网地址。这样在园区出口或者小运营商出口处，会有NAT的存在，所以总部与分支建立的VPN隧道必须能够支持NAT穿越。

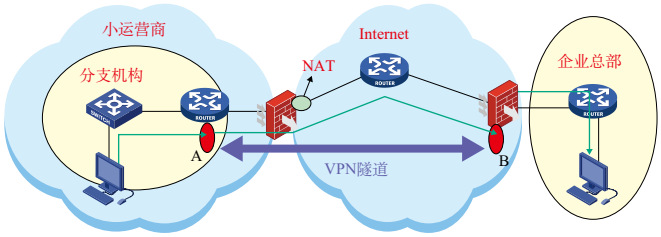


图1 VPN穿越NAT的应用组网

随着VPN技术得到广泛的应用，一些NAT技术与VPN技术存在不兼容的问题出现在我们面前。使NAT与VPN共存成为一个VPN实施中需要重点考虑的因素。本文主要介绍NAT与各种VPN共存可能引起的问题以及如何解决这些问题的方法。这里的VPN包括IPsec、GRE、L2TP、SSL VPN等常用传统VPN。由于NAT技术在转发报文的时候会修改报文的网络层地址和传输层端口，所以在分析传统VPN与NAT是否可以共存时，首先需要分析该VPN隧道的封装方式，看有没有传输层的端口，其次要分析VPN隧道的协商过程中是否要使用报文的IP地址。

IPsec与NAT

IPsec简介

因为在Internet早期，安全的需求非常少，在IP协议设计之初并没有考虑安全性，因此标准的IP协议是不安全的。随着Internet上的商业发展，安全问题日益突出，必须建立新的安全协议标准来满足这种需求，IPsec（IP Security）协议应运而生。

IPsec在三个方面保证了网络数据包的安全：机密性、完整性、认证性。机密性就是保证数据包的原始内容不被看到；完整性即保证数据包的内容不会被修改；认证性保证数据来自被信任的客户端。因此不可避免地会使用多种加密算法来修改数据包的内容。修改后的数据包有2种主要的封装形式：AH（Authentication Header）和ESP（Encapsulating SecurityPayload）。AH在IP数据包中插入了一个包头，其中包含对整个数据包内容的校验值。AH只用于对IP数据包的认证，而并不对数据包认证作任何修改。ESP用户加密整个数据包内容，同时也可以对数据包进行认证。AH和ESP可以同时使用也可以分开使用。

IPsec在传输数据的时候有2种不同的模式：传输模式和隧道模式。传输模式主要用于主机到主机之间的直接通信；而隧道模式主要用于主机到网关或网关到网关之间。传输模式和隧道模式主要在数据包封装时有所不同。在传输模式中，只有IP包的传输层部分被修改（认证或者加密）。而在隧道模式中，整个数据包包括IP头都被加密或认证。

在传输和隧道模式下的数据封装形式如图2所示，图中DATA为原IP报文数据。

工作模式 / 安全协议	传输模式	隧道模式
AH	原IP头 AH DATA	新IP头 AH 原IP头 DATA
ESP	原IP头 ESP DATA ESP-T	新IP头 ESP 原IP头 DATA ESP-T
AH-ESP	原IP头 AH ESP DATA ESP-T	新IP头 AH ESP 原IP头 DATA ESP-T

图2 IPsec VPN数据封装形式

IPsec与NAT的矛盾

在多数情况下NAT的处理对用户使用是完全透明的，但是当希望使用IPsec技术组建VPN网络时，NAT却带来了很大的麻烦。IPsec协议的主要目标之一是保护IP数据包的完整性，这意味着IPsec会禁止任何对数据包的修改。但是NAT处理过程是需要修改IP数据包的IP头数据、传输层报文头数据甚至传输数据的内容（如FTP应用），才能够正常工作。所以一旦经过IPsec处理的IP包

穿过NAT设备时，包内容被NAT设备所改动，修改后的数据包到达目的主机后其解密或完整性认证处理就会失败，于是这个报文被认为是非法数据而被丢弃。这就是组建VPN网关最常见的“IPsec与NAT协调工作”的问题。那么在什么时候我们会遇到这个问题呢？如果我们的VPN设备在NAT设备的后面，如果我们在外地上网需要通过VPN客户端访问公司内网，如果我们没有独立的公网地址，只能通过服务商接入Internet等等，都会遇到这个问题。

无论传输模式还是隧道模式，AH都会认证整个数据包。不同于ESP的是，AH还会认证位于AH头之前的IP头。当NAT设备修改了IP头之后，IPsec就会认为这是对数据包完整性的破坏，从而丢弃数据包。因此AH是绝对不可能和NAT在一起工作的。

我们再来看看使用ESP时的情况。ESP在传输模式时会保护TCP/UDP头，但是并不保护IP头，因此修改IP地址并不会破坏整个数据包的完整性。但是如果数据包是TCP/UDP数据包，NAT设备就需要修改数据包的校验值，而这个校验值是被ESP所保护的，这样却会导致完整性校验失败。 所以最终可能和NAT一起工作的只能是隧道模式下的ESP。

IKE和NAT工作时也同样存在着冲突。在IKE中第一阶段和第二阶段协商中会使用IP地址作为ID负载的内容。IKE报文穿越NAT后由于地址的改变会导致出现不一致的问题。IKE使用的源端口与目的端口号都是500。如果存在地址端口转换的情况，IKE报文的源端口将被改变，这样如果响应者判断源端口不是500，可能会存在协商问题。因此如果存在NAPT网关，要支持NAT穿越，必须支持非500端口发起的IKE报文，并且能够正确回应到这个非500端口。

IPsec和NAT求同存异

现在有许多的解决方案来解决NAT和IPsec共存问题，这里我们主要讨论一种最主要的解决方法：NAT-T和NAT穿越。NAT-T设计简单，不需要改动已有的设备或者协议，只需要边界设备支持即可。这个技术的基本思路是在IPsec封装好的数据包外再进行一次UDP的数据封装。这样，当此数据包穿过NAT网关时，被修改的只是最外层的IP/UDP数据，而对其内部真正的IPsec数据没有进行改动；在目的主机处再把外层的IP/UDP封装去掉，就可以获得完整的IPsec数据包。NAT-T在实际运作时，第一步是判断通信的双方是否支持NAT-T，这主要通过IKE协商时彼此发送的第一个数据包来判断。在判断双方均支持NAT-T后，进入到第二步，判断双方通信路径上是否存在NAT设备，这一步通常被称为NAT发现。NAT发现的原理实际上就是判断通信双发的IP地址或者端口是否发生了改变。当发现NAT设备以后，NAT-T双方开始协商所采用的数据包封装方式，至此完成协商过程。

在NAT发现步骤中感知到NAT设备存在的情况下，除了对数据包进行ESP处理外，还需要额外的进行一些其他的处理和封装。即对IPsec通信数据采用UDP和IKE端口的包头进行封装，因此这种封装后的数据包实际上和正常的IKE协商数据包有相同的包头（IP头和UDP头），从而避免防火墙对IPsec通信数据和IKE协商数据使用不同的安全策略。NAT-T后数据包的封装格式如图3、图4所示。

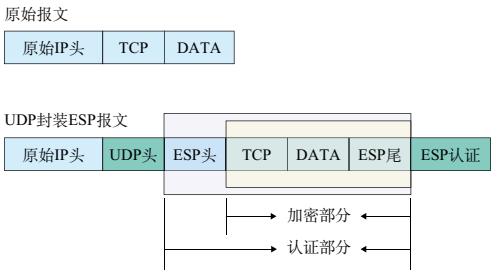


图3 UDP封装传输模式的IPsec对报文的封装

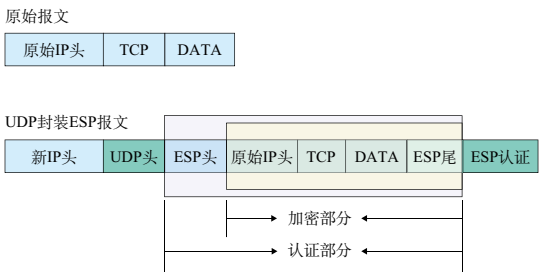


图4 UDP封装隧道模式的IPsec对报文的封装

通过在IP头后，ESP包头之前插入一个UDP包头来完成NAT-T的UDP封装。这样的数据包在经过NAT设备时就会被认为是一个正常的UDP数据包，从而完成NAPT转换，数据包到达VPN网关时也不会校验设备。显然，NAT-T加大了数据包的长度和负载，大约为每个数据包增加了20Byte的长度。这增加了设备处理的时间和负担。但是对应NAT-T的实现简单和安全而言，这是非常值得的，因此NAT-T实际上已成为VPN设备的必备功能。一些IKE报文敏感的NAT网关会对IKE进行特殊处理，针对该问题，采用了新的UDP端口号4500。这样支持NAT-T的设备不仅监听UDP 500，同时监听UDP 4500端口。

L2TP与NAT

L2TP简介

L2TP（Layer 2 Tunnel Protocol）称为二层隧道协议，是为在用户和企业的服务器之间透明传输PPP报文而设置的隧道协议，L2TP最大的优势在于充分利用了PPP协议的优势，提供了认证、地址分配等功能，非常适合远程用户或者分支机构通过Internet连接企业总部的私网。从某个角度来讲，L2TP实际上是一种PPPoIP的应用，就像PPPoE、PPPoA、PPPoFR一样，都是一些网络应用想利用PPP的一些特性，弥补本网络自身的不足。

为了支持L2TP这种PPPoIP的应用，各厂家在实现引进了虚接口去处理。一般都使用VT口作为一个配置管理的载体，VT如何具体去实现，各厂家在实现上有些细微的差别。

L2TP中定义了3个角色，CLIENT、LAC、LNS。LAC与LNS间是一个IP网络，LAC与CLIENT之间一般是一个PPP链路（常用的是PPPoE、DDR方式的PPP等）。L2TP的目的是在CLINET与LNS之间建立一条跨LAC的PPP链路，LAC透明传输PPP报文（封装到IP报文，具体是UDP）到LNS。

在L2TP应用中，PPP链路建立过程是这样的，首先是CLINET与LAC间进行LCP协商，一般接着会进行验证，验证通过后，LAC开始将验证报文等后续报文透明传送到LNS，也就是相当于在LNS与CLINET之间接着进行验证与IPCP协商，IPCP协商通过后，PPP链路就建立了。

L2TP与NAT可以共存

先看一下RFC2661对L2TP隧道端口号的描述：

L2TP使用已注册UDP端口1701【RFC1700】。整个L2TP包，包含负载和L2TP报头，被放在UDP报文中发送出去。L2TP隧道的初始化者选择一个可用的源UDP端口（可能是1701或者不是）然后发送到期望的目的地址的1701端口。接收者找到系统的一个空闲的端口（可能是或者不是1701端口），然后发送它的回复到初始化连接者的UDP端口和地址，发送的源端口号是找到的这个空闲的端口号。一旦源和目的端口还有地址建立连接以后，他们必须在隧道生命周期中保持稳定。上面建议指出接受者使用随意的端口号（和初始化隧道所用数据包不同的目的端口号，比如1701）将使得L2TP在穿越一些NAT设备的时候会更加困难，实现者在选择源端口的时候应该考虑这些隐含的冲突。

从上面这段话中可以看出为了穿越NAT，接受者已经不能使用随意的源端口号，而是必须使用1701端口了。

再来看一下L2TP VPN的封装格式，如图5所示，其存在UDP传输层的端口号，报文支持进行NAT穿越。

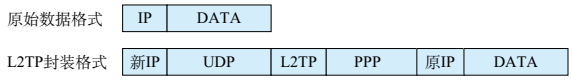


图5 L2TP VPN封装格式

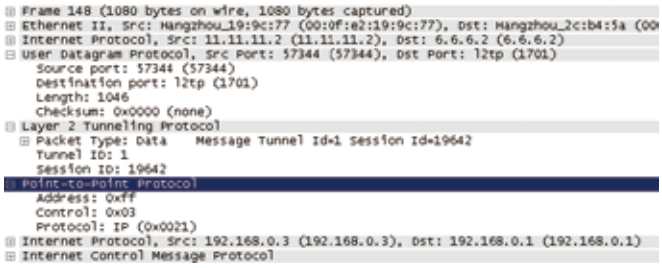


图6 L2TP VPN报文抓包

另外我们知道，L2TP实际上就是完成PPP Over IP的工作，L2TP首先需要建立L2TP Tunnel，然后在L2TP Tunnel上建立Incoming或者Outgoing session，最后建立PPP session，所有的L2TP需要承载的数据信息都是在PPP连接中进行传递的。

关于L2TP Tunnel、session和PPP的关系如图7所示：

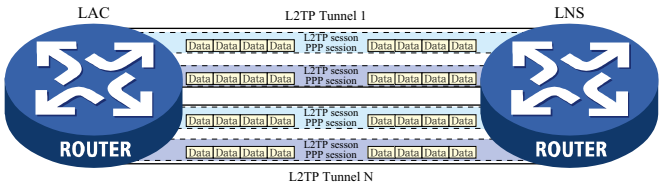


图7 L2TP Tunnel、session和PPP

■ L2TP Tunnel：用于建立L2TP控制连接，并且确定本地和远程的Tunnel ID，Tunnel ID是本地有效的。所有后续的L2TP报文都必须在建立了Tunnel以后在Tunnel中传输，建立L2TP控制连接需要完成检查Peer ID、检查Peer L2TP版本以及协商Peer的能力等过程。

■ L2TP session：可以是Incoming session（由LAC发起）或者Outgoing session（由LNS发起），L2TP session的主要作用是协商session ID（也是本地有效）、认证呼叫类型（Modem、ISDN）、认证呼叫信息（Calling Number、Called Number和Subaddress）等。

■ PPP session：LAC将Client的PPP信息通过L2TP session透传给LNS，使Client和LNS之间建立一个PPP session，透传的PPP信息包括LCP和NCP的所有参数，由LNS对Client进行认证以及地址的分配。

L2TP隧道协商过程中，没有使用到IP地址信息，可以说该协议是与NAT可以共存的。

GRE与NAT

GRE简介

GRE是对某些网络层协议（如：IP，IPX，AppleTalk等）的数据报文进行封装，使这些被封装的数据报文能够在另一个网络层协议（如IP）中传输，这是GRE最初的定义。最新的GRE封装规范，已经可以封装二层数据帧了，如PPP帧、MPLS等。在RFC2784中，GRE的定义是“X over Y”，X和Y可以是任意的协议。

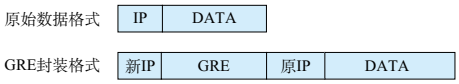
GRE隧道不能配置二层信息，但可以配置IP地址。GRE利用为隧道指定的实际物理接口完成转发，转发过程如下：

■ 所有发往远端VPN的原始报文，首先被发送到隧道源端；

■ 原始报文在隧道源端进行GRE封装，填写隧道建立时确定的隧道源地址和目的地址，然后再通过公共IP网络转发到远端VPN网络。

GRE与NAT誓不两立

我们首先来看一下GRE隧道的封装格式：



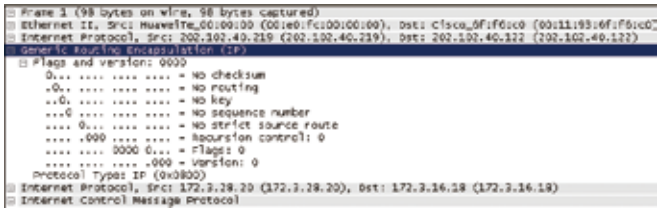


图8 GRE封装的报文

对于GRE的封装，从图8的抓包我们能清楚的看到，它没有传输层端口，不能进行NAPT转换。如果使用静态的NAT穿越，即使用一对一的NAT方式，GRE是可以支持的，但是由于这里所说的NAT设备都是指运营商网络中的，企业用户无法进行控制。另外GRE隧道两端也需要静态指定对端的公网IP，如果穿越NAT，报文的原IP地址发生改变后，隧道将无法匹配。因此在某种意义上，我们可以说GRE和NAT誓不两立，无法共存。

如果说用户一定要使用GRE VPN，且中间链路存在NAT时，一般通过VPN嵌套的方式来实现，譬如在GRE的隧道之上嵌套一层

IPsec VPN隧道，通过IPsec可以穿越NAT的特性来达到用户需求，这也是一类较为常见的解决方案。

SSL VPN与NAT

SSL VPN简介

SSL（Secure Sockets Layer，安全套接层）是一个安全协议，为基于TCP的应用层协议提供安全连接，如SSL可以为HTTP协议提供安全连接。SSL协议广泛应用于电子商务、网上银行等领域，为网络上数据的传输提供安全性保证。SSL VPN系统是一款采用SSL连接建立的安全VPN系统，为企业移动办公人员提供了便捷的远程接入服务。

SSL VPN的最大特色就是使用Web反向代理技术实现所谓的Web接入，使得用户可以只通过浏览器来访问内网资源。同时，为了适应已有的基于TCP/IP的应用程序，SSL VPN又提供了另外两种接入方式：端口转发和网络扩展。这样SSL VPN不但能做到对内部网络的全面访问，还能提供更安全、更方便的接入服务。

SSL VPN与NAT天然共存

SSL（Secure Socket Layer）安全套接层是一种运行在两台机器之间的安全通道协议；也可以运行在SSL代理和PC之间；其功能为保护传输数据（加密）和识别通信机器（认证）；SSL提供的安全通道是透明的，几乎所有基于TCP的协议稍加改动就可以直接运行于SSL之上。

不管SSL VPN运行在哪种模式下，都是以HTTPS（以SSL为基础的HTTP）为基础的VPN，VPN建立的过程和普通的TCP连接没有太大的区别，协议对NAT不敏感。

另外SSL VPN封装后的报文，为TCP的报文，存在传输层的端口，NAT对其没有任何的影响。

```
Frame 19 (1107 bytes on wire, 1107 bytes captured)
Ethernet II, Src: Elitegro_3f:49:da (00:16:ec:3f:49:da), Dst: Hangzhou_2c:b4:
Internet Protocol, Src: 136.1.35.168 (136.1.35.168), Dst: 4.4.4.1 (4.4.4.1)
    Version: 4
    Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 1093
    Identification: 0x7874 (30836)
  Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  Header checksum: 0xca90 [correct]
    Source: 136.1.35.168 (136.1.35.168)
    Destination: 4.4.4.1 (4.4.4.1)
Transmission Control Protocol, Src Port: 1564 (1564), Dst Port: https (443),
    Source port: 1564 (1564)
    Destination port: https (443)
    Sequence number: 340    (relative sequence number)
    [Next sequence number: 1393    (relative sequence number)]
    Acknowledgement number: 340    (relative ack number)
    Header length: 20 bytes
  Flags: 0x0018 (PSH, ACK)
    Window size: 64375
    Checksum: 0x6840 [correct]
Secure Socket Layer
  TLS Record Layer: Application Data Protocol: Hypertext transfer protocol
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 1048
      Application Data
```

图9 SSL VPN封装报文

总结

从上文中的分析，我们可以得出这样的结论，传统VPN中的SSL VPN和L2TP VPN与NAT天然可以共存，IPsec VPN在部分模式下可以与NAT共存，而GRE则无法穿越NAT网路。

NAT的双机热备方案

文/袁江

一般的NAT组网中，内网用户通过单台设备进行NAT转换访问外网，NAT设备承担了所有内外网之间的流量，无法规避单点故障。一旦发生单点故障，将导致内网用户无法与外网通信。

随着用户对网络可靠性的要求越来越高，发生单点故障导致网络间断是不可接受的。因此在重要节点处一般都部署两台或者多台设备，构成冗余备份组网，但如果设备之间不能实时的进行数据备份的话，链路切换时还是会导致用户的业务中断。双机热备方案可以很好避免该风险，该方案通过部署两台设备形成备份，通过VRRP或动态路由等机制进行链路切换，实现一台设备故障后流量自动切换到另一台正常工作的设备。

NAT的双机热备方案是两者的统一，同时实现内外网交互时的NAT功能及规避单点故障的双机热备功能，保证网络的不间断传输。

防火墙的双机热备

防火墙设备需要维护每条会话的状态等相关信息，当主设备故障、流量切换到备份设备时，要求备份设备上有正确的会话信息才能继续处理会话报文，否则会话报文会被丢弃从而导致会话中断。因此，主设备上会话建立或表项变化时需要将相关信息同步到备份设备，以保证主设备和备份设备会话表项的完全一致。防火墙需要同步的信息包括会话表、会话扩展信息、关联表等。数据同步的方式有批量备份和实时备份：

批量备份：防火墙设备工作了一段时间后，可能已经存在大量的会话表项，此时加入另一台防火墙设备，在两台设备上使能双机热备功能后，先运行的防火墙会将已有的会话表项一次性同步到新加入的设备，这个过程称为批量备份。

实时备份：防火墙在运行过程中，可能会产生新的会话表项。为了保证表项的完全一致，防火墙在产生新表项或表项变化后会及时备份到另一台设备，这个过程称为实时备份。

针对不同的组网环境，双机热备还实现了对非对称路径会话的备份。当备份类型为 不支持非对称路径备份时，一条会话中的数据流进入内网和从内网出去所经过的设备必须相同，即进入内网时经过双机热备中的一台设备，从内网出去时经过的设备是进入时经过的设备；当备份类型为支持非对称路径备份时，一条会话中的数据流进入内网和从内网出去所经过的设备可以不同，即进入内网时经过双机热备中的一台设备，从内网出去时经过的设备可以是进入时经过的设备，也可以是另一台设备。

根据组网情况，双机热备方案有两种工作模式：主备模式和负载分担模式（本文只描述了主备模式的NAT双机热备，负载分担模式中NAT的配置与主备模式相似）。

双机热备的两台设备间利用VRRP或动态路由实现流量的切换，流量模型如图1、图2所示。

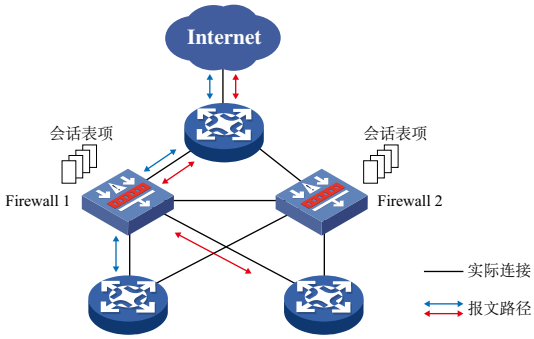


图1 主备模式下，Firewall 1故障前流量模型

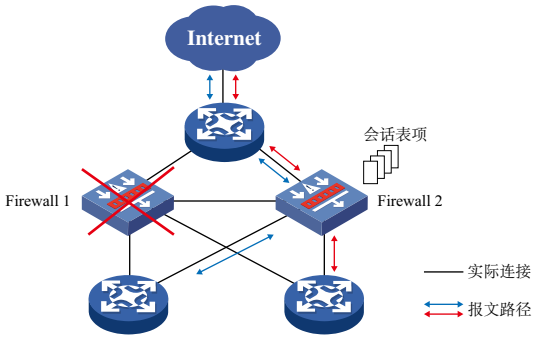


图2 主备模式下，Firewall 1故障后流量模型

NAT与双机热备的组合

地址池的低优先级属性

当双机热备的两台设备在网络中还需要完成NAT功能时，两台设备上配置的NAT地址池的地址空间必须完全一样，才能保证在一台设备发生故障时，另一台设备能够接替故障设备上的业务运行。但是，如果两台设备在做地址转换时，分别从各自的地址池中选用了相同的地址，且分配了相同的端口号，则会导致两台设备上的反向会话完全一样，无法进行会话数据的备份。

为解决该问题，NAT地址池引入了低优先级属性。在双机热备的两台设备上配置地址空间相同但优先级不同的地址池。例如：在两台设备上均配置地址池100.0.0.1~100.0.0.10，其中一台设备上的100.0.0.1~100.0.0.10地址池为低优先级。在进行地址转换时，低优先级NAT地址池中地址的端口取值范围为35001~65535，高优

先级地址池中地址的端口取值范围为1024~35000。这样主备两台防火墙虽然使用相同的NAT地址池中的地址，但是由于地址池的优先级不同，所以NAT转换后公网IP和公网端口就不会出现完全相同的情况了，在备份会话数据时就不会发生冲突。

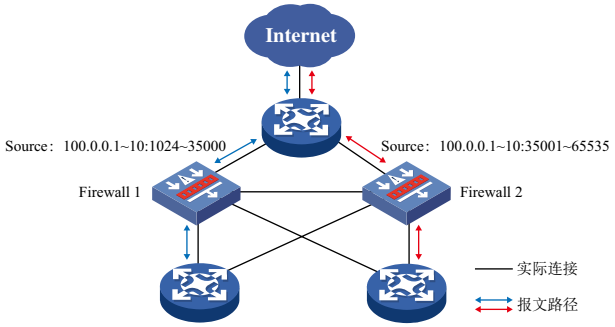


图3 负载分担模式下，NAT地址池的低优先级属性

低优先级地址池的不响应ARP请求属性

在特定组网条件下，双机热备支持NAT的两台设备还可能会发生ARP响应冲突的情况，如图4所示。Firewall 1和Firewall 2上均配置了NAT地址池100.0.0.1~100.0.0.10。当Router发起ARP请求，询问这两个地址池中的某个地址（如100.0.0.2），Firewall 1和Firewall 2都会收到这个ARP请求，并识别出这是自己地址池中的地址。这样，两台设备都会回复一个ARP响应，导致ARP响应冲突。为解决上述问题，NAT地址池引入了新的地址池ARP响应机制，即可以设置低优先级的地址池在设备的热备状态处于同步状态时不响应ARP请求。从而，保证了不会出现ARP响应冲突。

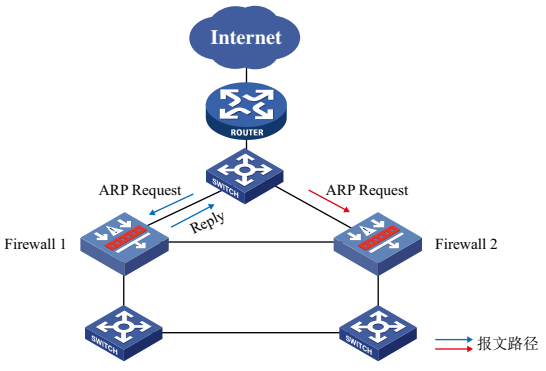


图4 低优先级地址池的不响应ARP请求属性

方案的实现

利用VRRP实现流量切换

通过VRRP将网络中的一组设备配置成一个备份组，这组设备在功能上就相当于一台虚拟设备。网络中的主机只需要知道这个虚拟设备的IP地址，通过这个虚拟设备与其它网络进行通信。备份组中，仅有一台设备处于活动状态，能够转发报文，称为主用设备（Master），其余设备都处于备份状态，并随时按照优先级高低做好接替任务的准备，称为备份设备（Backup）。当发现主用设备故障时，优先级次高的备用设备会当选为新的Master接替原Master工作，整个过程对用户来说是完全透明的，这就很好的实现了流量切换。

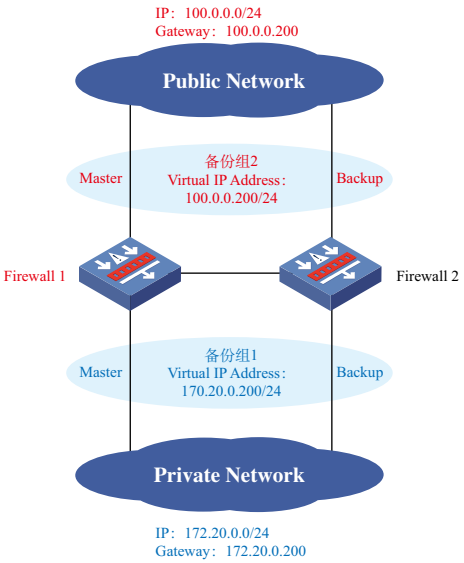


图5 通过VRRP实现流量切换（主备模式）

图5中连接私网和公网的两个接口分别配置备份组1和备份组2，私网的网关指定为备份组1的虚地址，公网的网关指定为备份组2的虚地址，当网络中主备切换时，上下游的网络设备是感知不到网络变化的，这样就确保了流量不间断。

NAT功能在公网侧的接口上开启，下面详细看看公网侧接口的配置：

Firewall 1：

```
NAT address-group 100 100.0.0.1 100.0.0.10 level 1  //地址池配置
#
acl number 2999                          //匹配所有报文
rule 0 permit
#
interface GigabitEthernet2/11
port link-mode route
NAT outbound 2999 address-group 100 track vrrp 2  //匹配ACL 2999规则的报文进行地址转换，地址池为100，关联VRRP备份组2
ip address 100.0.0.100 255.255.255.0
vrrp vrid 2 virtual-ip 100.0.0.200          //VRRP备份组2的虚地址
vrrp vrid 2 track interface GigabitEthernet2/10 reduced 10  //VRRP备份组2监控接口2/10（连接私网的接口）的状态，2/10的状态为Down或者Remove时，设备会主动降低VRRP备份组2的优先级
```

Firewall 2：ACL的配置相同

```
NAT address-group 100 100.0.0.1 100.0.0.10 level 0  //Level 0表示该地址池对应于双机热备中的低优先级地址池
#
interface GigabitEthernet2/11
 port link-mode route
 NAT outbound 2999 address-group 100 track vrrp 2
 ip address 100.0.0.101 255.255.255.0
 vrrp vrid 2 virtual-ip 100.0.0.200
 vrrp vrid 2 track interface GigabitEthernet2/10 reduced 10
```

双机热备的负载分担模式下利用VRRP切换流量的NAT配置与主备模式大致相同，本文档不做描述。

利用动态路由实现流量切换

如果网络中不同网段的两台设备A到B之间有多条通路，动态路由协议会使用算法选取最优的一条路径作为A到B的路由。当这条通路故障，路由协议会从其他可用通路中选择最优的一条作为新的路由，如果故障设备恢复，则会重新使用原路由，从而动态的保证A与B之间的连通。

双机热备的工作模式是主备模式还是负载分担模式可以通过组网和动态路由的配置来实现（以OSPF为例）：

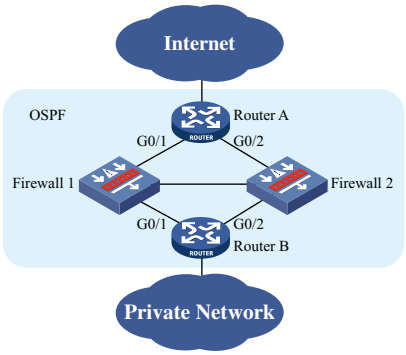


图6 通过动态路由实现流量切换

主备模式只有一台防火墙处于工作状态，另一台防火墙处于备份状态。如图6所示，Router A、Router B、Firewall 1和Firewall 2上均配置OSPF功能，处于同一个OSPF域，在Router A和Router B上都配置G0/1的cost值小于G0/2的。这样，路径Router A←→Firewall 1←→Router B的优先级会高于路径Router A←→Firewall 2←→Router B，当Firewall 1能正常工作的情况下，内网发往外网的报文都会通过Firewall 1转发；当Firewall 1发生故障，OSPF会启用次优路由，内网发往外网的报文会通过Firewall 2转发。

负载分担模式下两台防火墙处于工作状态并互为备份。如图6所示，Router A、Router B、Firewall 1和Firewall 2上均配置OSPF功能，处于同一个OSPF域，在Router A和Router B上都配置两条等价路由。因为Router A←→Firewall 1←→Router B这条路由与Router A←→Firewall 2←→Router B优先级一样，所以，当Firewall 1、Firewall 2能正常工作的情况下，Firewall 1和Firewall 2分担处理内网发往外网的报文；当Firewall 1发生故障，则Firewall 2会处理内网发往外网的全部报文。

NAT功能开启在公网侧的防火墙上，相关配置如下：

Firewall 1：

```
ospf 1
import-route static              //引入NAT地址池中地址的路由，发布给上行设备
area 0.0.0.100
network 192.168.0.0 0.0.255.255     //私网侧的地址网段
network 100.0.0.0 0.0.255.255       //公网侧的地址网段
#
```

```
NAT address-group 100 100.0.0.1 100.0.0.10 level 1
#
acl number 2999
rule 0 permit source 100.0.0.0 0.0.0.255
#
interface GigabitEthernet2/10          //连接私网的接口
port link-mode route
ip address 192.168.0.100 255.255.255.0
#
interface GigabitEthernet2/11          //连接公网的接口
port link-mode route
NAT outbound 2999 address-group 100
ip address 100.0.0.100 255.255.255.0
```

Firewall 2：

```
ospf 1
import-route static
area 0.0.0.100
network 192.168.0.0 0.0.255.255
network 100.0.0.0 0.0.255.255
#
NAT address-group 100 100.0.0.1 100.0.0.10 level 0
#
interface GigabitEthernet2/10          //连接私网的接口
port link-mode route
ip address 192.168.1.100 255.255.255.0
#
interface GigabitEthernet2/11          //连接公网的接口
port link-mode route
NAT outbound 2999 address-group 100
ip address 100.0.1.100 255.255.255.0
```

小结

NAT的双机热备方案，解决IPv4地址短缺问题的同时也很好的提高了组网系统的可靠性，拓展了NAT的应用场景，为NAT在多种环境下的应用提供了解决方案。

NAT典型组网实例

文/曹佐清 张岩 贾欣武 王军

NAT多实例

功能说明

多实例的概念是指路由的多实例，每一个实例之间路由相互隔离。通常情况下，每一个VPN看作为一个实例。

NAT多实例允许分属于不同实例的用户通过同一个出口访问外部网络，同时允许分属于不同实例的用户使用相同的私网地址。当实例内的用户访问外部网络时，地址转换将内部网络主机的IP地址和端口替换为设备的外部网络地址和端口，同时还记录了用户的实例信息（如MPLS VPN协议类型和路由标识符RD等）。回应报文到达时，地址转换将外部网络地址和端口还原为内部网络主机的IP地址和端口，可得知是哪一个实例用户的访问。

同时，地址转换支持内部服务器的多实例，给外部提供访问实例内主机的机会。例如，MPLS VPN1内提供Web服务的主机地址是10.110.1.1，可以使用202.110.10.20作为Web服务器的外部地址，Internet的用户使用202.110.10.20的地址就可以访问到MPLS VPN1提供的Web服务。

另外，NAT还可利用外部网络地址所携带的实例信息，支持多个实例之间的互访。

组网需求

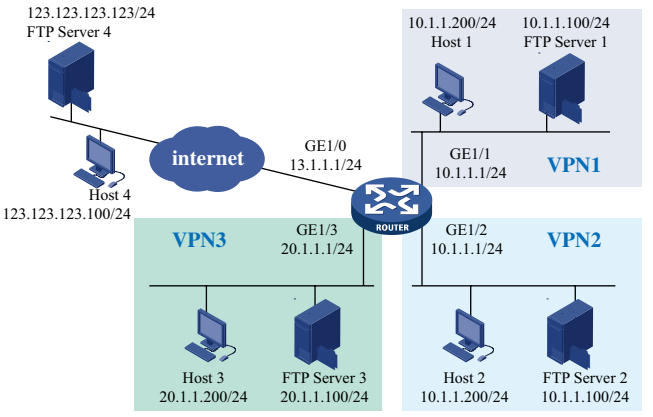


图1 NAT多实例组网

一个公司拥有13.1.1.1/24至13.1.1.4/24四个公网地址。公司内部划分了三个部门，彼此隔离到三个不同的VPN内，每个VPN可以看作一个实例。VPN1和VPN2的两个部门规划的网络内部IP相同。

■ 两个VPN内【VPN1、VPN2】的10.1.1.200和10.1.1.100的用户可以访问Internet，其它用户不能访问Internet。两个VPN内用户访问Internet使用的公网地址为13.1.1.2；

■ 公网用户Host 4可以访问VPN 【VPN1】内的FTP Server，Host 4通过13.1.1.3访问FTP Server 1；

■ VPN3内的Host 3通过20.1.1.222地址，可以访问VPN2的FTP Server 2。

实现配置

■ 按照拓扑完成IP地址等配置信息，完成VPN的设置，以及公网的路由设置；

■ 需要在两个VPN【VPN1、VPN2】内静态路由方式设置到达公网FTP Server 4的下一跳IP，确定公网的出接口；

■ 根据需求完成VPN内用户是否允许经过NAT的权限设置，添加ACL匹配两个MPLS VPN内的地址10.1.1.100/32、10.1.1.200/32；

```
参考命令行
【Router】acl number 2001
【Router-acl-basic-2001】rule permit VPN-instance VPN1 source 10.1.1.100 0
【Router-acl-basic-2001】rule permit VPN-instance VPN1 source 10.1.1.200 0
【Router-acl-basic-2001】rule permit VPN-instance VPN2 source 10.1.1.100 0
【Router-acl-basic-2001】rule permit VPN-instance VPN2 source 10.1.1.200 0
【Router-acl-basic-2001】rule deny
【Router-acl-basic-2001】quit
```

■ 建立NAT转换地址池，使用13.1.1.2；

```
参考命令行
<Router> system-view
【Router】NAT address-group 1 13.1.1.2 13.1.1.2
```

■ 在出口GE1/0上应用基于ACL的NAT出方向转换；

```
参考命令行
    【Router】interface gigabitethernet 1/0
    【Router-GigabitEthernet1/0】NAT outbound 2001 address-group 1
```

此时可以实现VPN内的用户访问公网FTP Server 4的功能。每个NAT表项内，会有标明VPN的RD信息，用以区分相同网段的不同VPN用户；

■ 在公网出接口GE1/0上，设置映射内部服务器的命令；命令中需要的元素有FTP Server对外映射的公网IP地址13.1.1.3/32、提供的服务为FTP、VPN内部FTP Server的IP地址、以及所在VPN信息。设置后，Host 4就可以‘ftp 13.1.1.3’方式登陆FTP Server 1；

```
参考命令行
【Router】interface gigabitethernet 1/0
【Router-GigabitEthernet1/0】NAT Server protocol tcp global 13.1.1.3 ftp inside 10.1.1.100 ftp VPN-instance VPN1
```

■ VPN3内的Host 3需要访问VPN2的FTP Server 2，首先需要在VPN3的接口GE1/3上设置NAT内部Server相关的命令，此命令与上面类似。不过这次增加了虚拟Server IP地址所在的VPN信息；

```
参考命令行
【Router】interface gigabitethernet 1/3
【Router-GigabitEthernet1/3】NAT Server protocol tcp global 20.1.1.222 ftp VPN-instance VPN3 inside 10.1.1.100 ftp VPN-instance VPN2
```

■ VPN3中配置到达VPN2中FTP Server 2的路由，VPN2中配置到达VPN3中Host 3的路由。

此时完成相关配置，Host 3可以通过‘ftp 20.1.1.222’的方式访问FTP Server 2。

SR8800采用多实例实现地址冲突情况下两次NAT应用

功能说明

两次NAT：一个NAT变换能够同时实现源地址的变换和目的地址的变换。

双向NAT：能够从内网和外网同时发起NAT变换的首包流程，也就是既能实现内网对外网的访问，也能够同时实现外网对内网资源的访问。

组网需求

某银行需要引入外联单位进入内网，为了不暴露自身私网细节以及对外联网络的统一管理，要求将内网向外联单位访问的数据流量先NAT至对外统一网段100.0.0.0/16后再转发至相关外联单位，且要将外联单位向内网访问的数据流量先NAT至银行为其分配的外联单位网段后再进入内网，使得银行内网路由可统一管理。银行为外联单位1分配的网段为110.0.0.0/16，为外联单位2分配的网段为120.0.0.0/16。由于外联单位较多且都独自规划各自内网，在实施时出现一个外联单位和银行内网网段重叠情况，均为10.0.0.0/16。银行客户要求在出口设备上解决该地址重叠问题，并需保证各本来不互通的外联单位仍保证网络访问隔离，所有的NAT操作也只在其一台出口路由器SR8800上进行。具体组网图见图2。

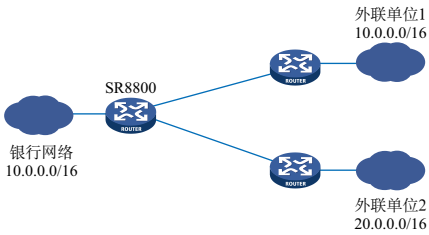


图2 两次NAT组网需求图

实现分析

实现难点

对于边界设备来说，两个接口都有相同网段的路由10.0.0.0/16，出现路由上的冲突，设备该如何将数据送到正确的出口进行转发；按照用户需求，银行与外联单位进行业务互通时，在银行出口设备上需要对源和目的均进行转换，要求设备能够同时转换源和目的IP，即具有两次NAT的功能；并需保证本来不互通的各外联单位仍保证网络访问隔离。

实现方式

■ 将接口绑入VPN实现重叠路由隔离，NAT后的目的地址跨VPN寻址；

■ 利用静态NAT跨VPN转换实现源和目的地址的两次转换；

■ 配置路由策略实现外联单位间的路由隔离。

实现配置

■ 配置VPN1、VPN2和VPN3分别用于绑定连接银行、外联单位1、外联单位2的物理接口，配置VPN NAT作为两次NAT的中转VPN。配置VPN NAT的export target，配置VPN1，VPN2和VPN3的import target与之匹配；

```
#
ip VPN-instance VPN1
description "Bank"
route-distinguisher 1:1
VPN-target 10:1 import-extcommunity
#
ip VPN-instance VPN2
description "Company1"
route-distinguisher 2:2
VPN-target 10:1 import-extcommunity
#
ip VPN-instance VPN3
description "Company2"
route-distinguisher 3:3
VPN-target 10:1 import-extcommunity
#
ip VPN-instance NAT
description "ForNatAddress"
route-distinguisher 10:1
VPN-target 10:1 export-extcommunity
#
```

■ 在NAT板卡视图下配置NAT static net-to-net命令，将源VPN1，VPN2，VPN3的地址网段静态转换为VPN NAT实例下的分配地址网段，如将VPN1的10.0.0.0/16网段转换为VPN NAT的100.0.0.0/16网段；

```
#
interface NAT1/0/1
NAT static net-to-net 10.0.0.0 VPN-instance VPN1 100.0.0.0 VPN-instance NAT 255.255.0.0
NAT static net-to-net 10.0.0.0 VPN-instance VPN2 110.0.0.0 VPN-instance NAT 255.255.0.0
NAT static net-to-net 20.0.0.0 VPN-instance VPN3 120.0.0.0 VPN-instance NAT 255.255.0.0
#
```

■ 启用BGP，在IPv4-family VPN-instance NAT视图下引入设备自动生成的NAT后网段静态路由；

```
#
bgp 100
ipv4-family VPN-instance NAT
import-route static
#
```

■ 配置两个ACL，第一个用于匹配路由110.0.0.0/16和120.0.0.0/16，第二个用于匹配路由100.0.0.0/16。配置两个route-policy 1和2，均只配置一个permit节点，route-policy 1匹配第一个ACL，route-policy 2匹配第二个ACL。再在银行侧的IP VPN-instance VPN1视图下配置import route-policy 1，在外联单位侧ip VPN-instance VPN2和VPN3视图下配置import route-policy 2。如此实现不同外联单位间的路由隔离。

```
#
acl number 2000
rule 0 permit source 110.0.0.0 0.0.255.255
rule 5 permit source 120.0.0.0 0.0.0.255
acl number 2001
rule 0 permit source 100.0.0.0 0.0.255.255
#
route-policy 1 permit node 10
if-match acl 2000
route-policy 2 permit node 10
if-match acl 2001
#
ip VPN-instance VPN1
import route-policy 1
#
ip VPN-instance VPN2
import route-policy 2
#
ip VPN-instance VPN3
import route-policy 2
#
```

NAT实现Load Balance应用

负载均衡概述

LB（Load Balance，负载均衡）是一种集群技术，它将特定的业务（网络服务、网络流量等）分担给多台网络设备（包括服务器、防火墙等）或多条链路，从而提高了业务处理能力，保证了业务的高可靠性。

负载均衡分类

负载均衡包括服务器负载均衡、防火墙负载均衡和链路负载均衡三种类型，三种负载均衡的应用场景如下：

■ 服务器负载均衡：在数据中心等组网环境中，可以采用服务器负载均衡，将网络服务分担给多台服务器进行处理，提高数据中心的业务处理能力；

■ 防火墙负载均衡：在防火墙的处理能力成为瓶颈的组网环境中，可以采用防火墙负载均衡，将网络流量分担给多台防火墙设备，提高防火墙的处理能力；

■ 链路负载均衡：在有多个运营商出口的组网环境中，可以采用链路动态负载均衡，实现链路的动态选择，提高服务的可靠性。

NAT方式的服务器负载均衡

下面介绍与NAT相关的服务器负载均衡的工作机制。NAT方式的负载均衡是指通过利用NAT技术对报文的源、目的地址进行转换，使客户端和各个真实服务器之间进行通信。

NAT方式的服务器负载均衡组网图3所示：

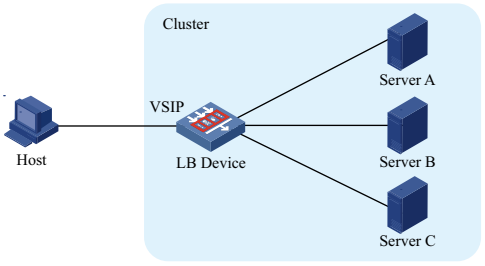


图3 NAT方式的服务器负载均衡组网图

NAT方式的服务器负载均衡包括以下几个基本元素：

■ 集群（Cluster）：对外提供特定服务的群体，包括LB Device和Server；

■ LB Device：负责分发各种服务请求到多台Server的设备；

■ Server：负责响应和处理各种服务请求的服务器；

■ VSIP：集群对外提供的虚服务IP（Virtual Service IP），供用户请求服务时使用；

■ Server IP：服务器的IP地址，供LB Device分发服务请求时使用。

NAT方式服务器负载均衡的工作流程如图4所示：

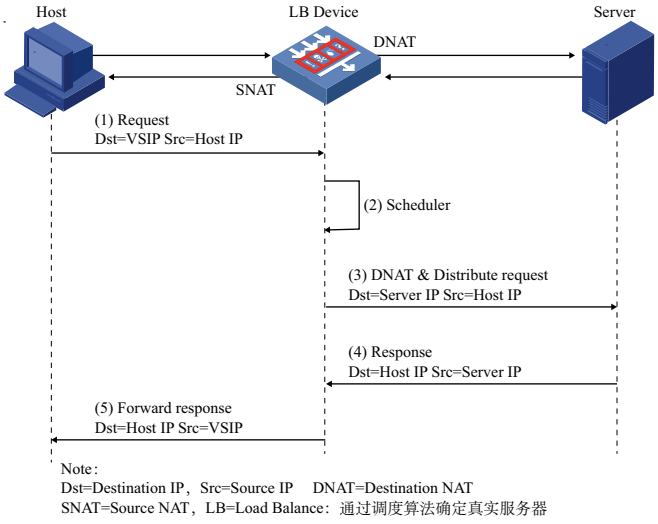


图4 NAT方式的服务器负载均衡流程图

下面简要描述单向NAT转换的报文交互流程：

■ Host发送服务请求报文，报文的源IP为Host IP，报文目的IP为VSIP；

■ LB Device接收到请求报文后，借助调度算法计算出应该将请求分发给哪台Server；

■ LB Device使用DNAT（Destination NAT，目的地址NAT）技术分发报文，报文的源IP为Host IP，报文目的IP为Server IP；

■ Server接收并处理请求报文，返回响应报文，报文的源IP为Server IP，报文目的IP为Host IP；

■ LB Device接收响应报文，转换源IP后转发，报文的源IP为VSIP，报文目的IP为Host IP。

NAT方式服务器负载均衡具有如下特点：

■ 上述LB Device是作为单向NAT设备进行地址转换，LB Device还可以进行双向NAT转换，即LB Device选择服务器时，使用DNAT和SNAT技术对报文的目的地址和源地址都进行了转换；

■ 所有请求和响应报文都经过LB进行转发，流程可控，便于扩展安全功能，这是优点，但LB Device负载较重，容易成为性能瓶颈；

■ 组网灵活，对服务器没有额外要求，不需要修改服务器配置，适用于各种组网。

NAT与DNS的组合应用

DNS ALG

通常情况下，NAT只对报文头中的IP地址和端口信息进行转换，不对应用层数据载荷中的字段进行分析。然而一些特殊协议，它们报文的数据载荷中可能包含IP地址或端口信息，这些内容不能被NAT进行有效的转换，就可能导致问题。ALG（Application Level Gateway，应用层网关）主要完成对应用层报文的处理，主要用来解决在NAT环境中只转换IP地址和端口信息，无法有效处理数据载荷的问题。

如下组网图5中，如果NAT网关没有开启ALG，PC1访问www服务器就会出现问题：

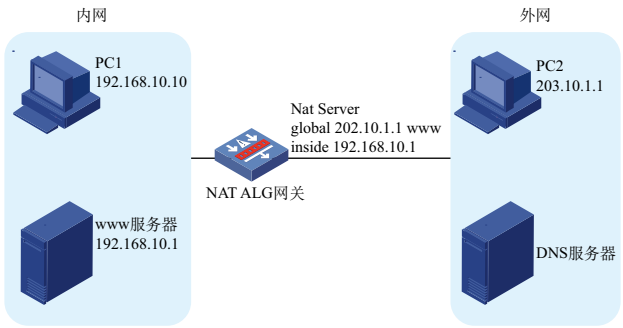


图5 NAT DNS处理组网图

没有开启DNS ALG的情况下，处理流程如下：

■ PC1访问www服务器，向外网的DNS服务器发送DNS解析请求，源IP为192.168.10.10，目的IP为DNS服务器地址；

■ DNS请求经过NAT网关转换源地址后发送给DNS服务器；

■ DNS服务器对PC1的请求进行响应，响应内容包括www域名与IP地址的对应关系，注意，外网DNS服务器上www域名对应的IP地址是NAT网关上配置的外网地址202.10.1.1；

■ NAT网关收到响应报文后转换目的地址发向PC1；

■ PC1收到该响应报文后向外网地址202.10.1.1发起数据访问，而真实的服务器在内网，显然这样的访问不会成功。

开启DNS ALG后的处理流程：

■ NAT网关收到DNS响应报文，由于开启了DNS ALG功能，所以NAT网关会监听DNS服务器的响应报文，在本例中由于外网接口上配置的NAT Server公网地址为202.10.1.1，返回的DNS响应报文中域名对应的地址也是202.10.1.1，所以NAT网关会修改DNS响应报文，将DNS响应报文中域名对应的地址修改为内网服务器真实地址192.168.10.1，同时将响应报文的目的IP进行转换目的地址后发向PC1；

■ PC1收到该响应报文后向192.168.10.1发起数据访问，由于真实的服务器就在内网，所以访问成功。

DNS Mapping

DNS Mapping主要用于解决普通的NAT ALG在特定组网时的问题，如在图6中：DNS服务器在公网上，NAT网关上使用一个公网地址映射内部的多个Server，内网PC1需要通过域名正常访问内网服务器。

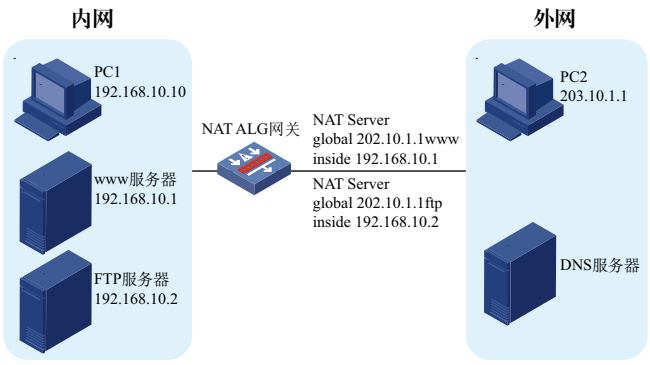


图6 DNS Mapping组网图

问题现象描述

如果没有开启DNS Mapping特性

■ PC1访问www服务器和FTP服务器，向外网的DNS服务器发送两份DNS请求，源IP为192.168.10.10，目的IP为DNS服务器地址；

■ 两份DNS请求经过NAT网关转换源地址后发送给DNS服务器；

■ DNS服务器对PC1的请求进行响应，两份响应内容分别为：www域名对应地址202.10.1.1，FTP域名对应地址202.10.1.1，注意DNS响应报文中不包括端口信息；

■ NAT网关收到响应报文，由于开启了DNS ALG功能，所以NAT网关会监听DNS服务器的响应报文，在本例中由于外网接口上配置的NAT Server公网地址为202.10.1.1，返回的DNS响应报文中域名对应的地址也是202.10.1.1，所以需要修改DNS响应报文，将DNS响应报文中域名对应的地址修改为内网服务器真实地址，但问题出现了，由于DNS服务器发回的DNS响应报文中不包括端口信息，所以网关在修改DNS响应报文时，有可能将内部真实的服务器地址填写错误，如：将FTP域名对应的服务器地址也填写成192.168.10.1；

■ PC1收到该响应报文后向192.168.10.1发起FTP访问，但实际开启FTP服务的实际是192.168.10.2，所以FTP访问不会成功。

DNS Mapping原理

配置了NAT的接口上收到DNS响应报文后会根据报文中的域名查找用户配置的映射表，如果存在对应的表项，则根据表项内的“公网地址—公网端口—协议类型”查找内网服务器，如果找到指定的内部服务器，则用服务器的私网地址替换DNS查询结果中的公网地址。如果没有配置此域名或对应的内部服务器不存在，则按原来流程替换。然后正常转发此DNS响应报文。这样就能避免前面提到的多个内网服务器对应一个公网地址时出现问题。

使能DNS Mapping后的处理流程

在NAT网关上配置DNS Mapping：

```
NAT dns-map domain www.com protocol tcp ip 202.10.1.1 port www
NAT dns-map domain ftp.com protocol tcp ip 202.10.1.1 port ftp
```

加载上述配置后NAT网关的处理流程如下：

NAT网关收到DNS服务器返回的DNS响应报文，载荷中域名FTP.com对应IP地址202.10.1.1，NAT网关查找dns-map表，找到匹配的表项，得到公网地址—域名—公网端口—协议类型的对应关系，然后再从接口的NAT Server配置中得到对应的内网FTP服务器地址192.168.10.2，NAT网关将DNS响应报文中的地址修改为

192.168.10.2发向PC1，PC1收到后向192.168.10.2发起FTP访问，访问正常；PC1如果向www.com发起访问，处理方式也类似，能够解析到对应内网www服务器的IP地址，不再赘述。

在dns-map表中为何不直接指定私网服务器的IP地址?

有两个原因：

■ 在实际组网中，对外提供的NAT Server公网地址可以认为是固定的（外网用户只看到公网地址，并且在DNS Server上记录的也是域名与公网地址的映射），但内网服务器的地址未必是固定的，如果内网服务器地址变化，dns-map表不需变化，仅修改接口NAT Server的配置即可；

■ 在有些组网中，需要提供服务器负载均衡功能，即：对外表现为一个NAT Server外网服务器，内网可以有多个内网服务器，实际浏览可以根据算法重定向到不同内网服务器。这种情况下dns-map表不需变化，在NAT Server的配置中关联Server-group即可。

地址重叠情况下NAT与DNS的组合应用

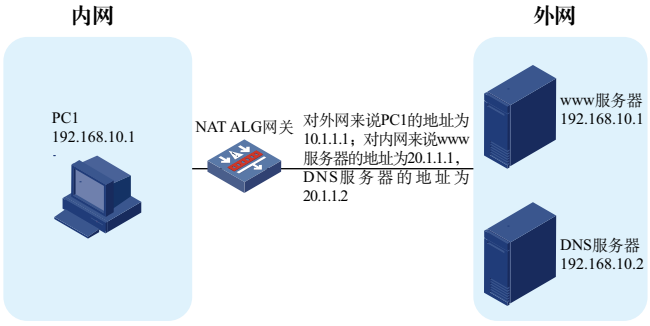


图7 地址重叠情况下NAT与DNS组合应用

在图7所示的场景中，内外网地址完全重叠，需要双向NAT（SNAT、DNAT）与DNS ALG组合才能实现正常访问，流程简述如下：

■ PC1向www服务器发起访问，首先发起DNS查询，源IP为192.168.10.1目的IP为20.1.1.2；

■ 查询报文到达NAT网关，网关经过双向NAT处理后发向DNS Server的报文源IP为10.1.1.1，目的IP为192.168.10.2，源和目的IP全部改变；

■ DNS Server收到查询后向NAT网关发出响应报文，源IP为192.168.10.2，目的IP为10.1.1.1，DNS载荷为www服务器对应的IP地址192.168.10.1；

■ NAT网关收到后匹配双向NAT建立的session并且经过DNS ALG处理，发向PC1报文的源IP为20.1.1.2，目的IP为192.168.10.1，DNS载荷变成20.1.1.1；

■ PC1向www服务器发起连接，源IP为192.168.10.1，目的IP为20.1.1.1；

■ www数据报文到NAT网关，网关经过双向NAT处理后发向www服务器的报文源IP为10.1.1.1，目的IP为192.168.10.1，源和目的IP全部改变；

■ www回程报文的地址转换流程与DNS报文处理类似，不再详述。

注：地址重叠场景下的NAT与ALG处理在H3C各产品上实现及支持情况可能有差异，请以具体产品实现为准。

NAT与DDNS的组合应用

我们知道，DNS域名比IP更容易被记住，所以服务器一般会有自己的域名。但对于一个使用NAT来连接公网的企业或组织，仅有的出口公网IP也可能是动态分配的，静态配置的DNS无法适应这种场景。此时就会用到动态DNS技术，即DDNS。，DDNS协议用于这种情况下对域名与IP地址的对应关系进行及时更新，DDNS客户端会将变化后的结果及时更新给服务器，DNS客户可以按照正常的DNS查询来得知这种变化。DDNS特性与NAT特性共同部署时的组网图见图8：

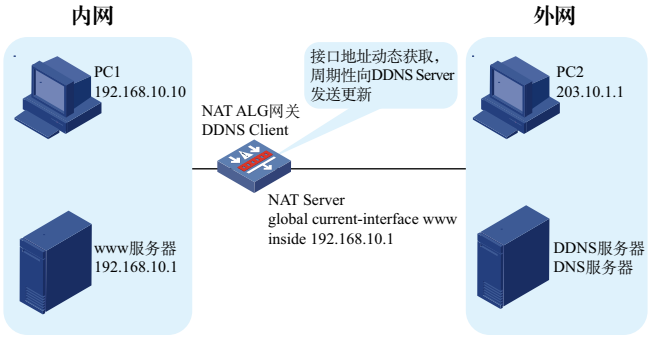


图8 NAT环境下的DDNS

组网描述

内网服务器位于NAT网关之后，需要配置NAT Server，同时为了使内外网上的所有用户都可以访问内网服务器提供的服务，NAT Server又必须采用对端给本端接口动态分配的公网地址，这导致在DNS服务器上域名对应的地址是动态变化的，因此在NAT网关上需要采用DDNS更新报文将最新的域名与IP对应关系发送给DDNS Server。其中NAT网关同时充当DDNS client角色，负责将外网口地址（同时也是NAT Server的global地址）与域名的对应关系及时向DDNS Server进行更新。

处理过程描述

■ NAT网关的外网地址更新后向DDNS Server发送DDNS更新报文，假设对端给NAT网关分配到的地址为202.10.10.10；

■ DDNS Server收到后对域名www.com与IP的对应关系进行更新；

■ PC1访问www服务器，向外网的DNS服务器发送请求，源IP为192.168.10.10，目的IP为DNS服务器地址；

■ DNS请求经过NAT网关转换源地址后发送给DNS服务器；

■ DNS服务器对PC1的请求进行响应，响应内容包括www域名与IP地址的对应关系，由于服务器中的域名与IP地址对应关系已经经过更新，所以回应的DNS响应报文中www.COM对应的地址为202.10.10.10；

■ NAT网关收到DNS响应报文后检查该消息，DNS ALG发现当前NAT Server地址与收到的DNS响应报文中的地址相同，就是动态获取的接口地址，所以需要修改DNS响应报文，将DNS响应报文中域名对应的地址修改为内网服务器的真实IP地址192.168.10.1，同时将响应报文的目的IP进行转换目的地址后发向PC1；

■ PC1收到该响应报文后向www服务器的内网地址192.168.10.1发起访问，访问成功。

注：目前，DDNS更新过程没有统一的标准，向不同的DDNS服务器请求更新的过程各不相同。

总结

由上面的步骤可以看出，对于内网的PC1，在DNS Server返回的响应报文中域名与IP地址对应关系动态变化的情况下，NAT网关在ALG的协助下进行载荷修改，内网的PC1总是能够得知内网服务器的真实IP地址，可以直接发起访问；

对于外网的PC2，能够直接收到DNS Server发送的DNS响应报文，因此发起的真实数据访问可以根据DNS响应的内容进行动态变化。

DNS客户端向DDNS Server发起DNS查询，查询过程是标准的DNS查询，流程与DNS客户端向普通DNS Server发起查询过程完全相同，如果出现与图5、图6类似的组网需要用到前述介绍过的ALG和DNS Mapping特性，处理流程与前述流程类似。

NAT-PT介绍

文/田浩博

前言

NAT-PT和NAT的差别

如今，随着全球的IPv4地址已经分配完毕，再也没有新的空闲的IPv4地址可以分配了，这更加让IPv4地址的使用成为一种奢侈。NAT（Network Address Translation，网络地址转换）在IPv4地址严重不足的情况下提出缓解办法，是将IP数据报文头中的IP地址转换为另一个IP地址的过程。在实际应用中，NAT使一个局域网中的主机使用少量合法的IPv4地址就可以访问外部资源。这种通过使用少量的公网IP地址代表较多的私网IP地址的方式，将有助于减缓可用IP地址空间的枯竭。

不过，NAT不能永久的解决当今IP地址短缺的问题，因此IPv6的应用越来越显得重要。IPv6的应用是个循序渐进的过程，在很长时间内，IPv4网络和IPv6网络会同时存在且需要相互通信。基于这种情况，就需要一种技术，解决IPv4和IPv6网络的互通问题，所以NAT-PT技术应运而生。

NAT-PT（Network Address Translation-Protocol Translation，附带协议转换的网络地址转换）技术秉承NAT技术（RFC2663）的思想，但在原理方面大有不同，可以这样简单的理解，NAT技术是IPv4私网地址与公网地址之间的转换，它是为了解决IPv4公网地址缺乏问题；NAT-PT技术则是IPv6协议与IPv4协议之间的转换，在RFC 2765与RFC 2766中给出了其定义它是为了解决两者的互通问题。在IPv4网络完全过渡到IPv6网络之前，两种类型网络之间直接的通信可以通过NAT-PT来实现。

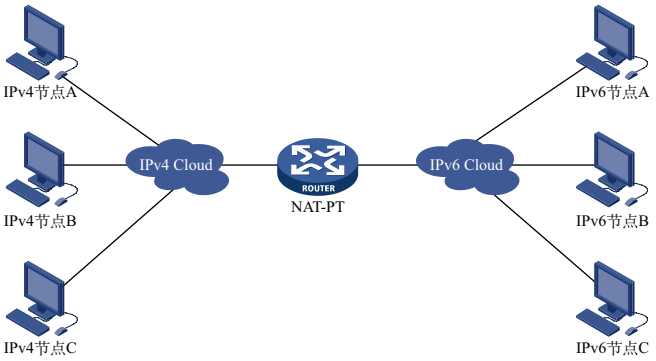


图1 NAT-PT示意图

如图1所示，NAT-PT作用于IPv4和IPv6网络边缘设备上，所有的地址转换过程都在该设备上实现，对IPv4和IPv6网络来说是透明的，即用户不必改变目前的IPv4网络中主机的配置就可实现IPv6网络与IPv4网络的通信。

NAT-PT和NAT本质的区别在于应用场合的不同。NAT是IPv4网络中公网、私网地址的相互转换，而NAT-PT则是IPv6和IPv4地址的相互转换，加入了IPv6和IPv4间协议的转换内容。比如对于ICMP报文的处理上，所有的ICMPv4报文向ICMPv6报文转换的时候都必须要重新计算校验和，因为ICMPv6像TCP和UDP一样有一个伪报头的检验和。另外所有Type字段都需要进行更新，如果是ICMP差错报文的话，那么差错信息中的IP头也同时需要转换。这些报文头转换的同时很可能引起报文长度的变化，因此IPv6报文最外层净载荷长度字段必须被更新。同样，ICMPv6向ICMPv4进行转换也是类似的流程。

NAT-PT和隧道的差别

NAT-PT和隧道都是IPv4向IPv6过渡的技术，但是它们之间的实现和适用范围都是有很大的不同的。

首先报文转换和转发的方式不同，这个是NAT-PT和隧道最根本的差别。NAT-PT是对报文的网络层内容进行转换修改，剥离原先的报文头，替换为转换之后的报文头（IPv4→IPv6或者IPv6→IPv4）；而隧道是对初始报文作另一层报文封装，根据隧道的不同类型加上相应的报文头。

正是由于转换的方式不同，导致他们检查报文的方式也不同。NAT-PT会检查并且可以更改报文中的端口号；而隧道从来不会检查报文的传输层内容。

转换方式的不同同样致使了他们的使用范围不一样，NAT-PT一般适用于IPv4与IPv6不同网络中主机互相访问的环境中；而隧道一般用于实现一种网络协议跨越另一种网络协议之间的通讯。所以，形成NAT-PT的环境只需要有一台可以进行NAT-PT转换的设备即可；而构造一种隧道的环境就必须要两台设备形成一个隧道。

NAT-PT的基本原理

NAT-PT前缀

不论采用哪种NAT-PT机制，配置NAT-PT前缀都是必须的。NAT-PT前缀是长度为96位的IPv6地址前缀，它具有以下两个作用：

- 从IPv6网络发送到IPv4网络的报文到达NAT-PT设备后，设备会检测报文目的IPv6地址的前缀，只有与所配置的NAT-PT前缀相同的报文才允许进行IPv6到IPv4的转换；
- 从IPv4网络发送到IPv6网络的报文，经过NAT-PT转换后，源IPv6地址的前缀为配置的NAT-PT前缀。

NAT-PT机制及原理

类似于NAT，NAT-PT有几种机制可实现IPv4和IPv6地址之间的相互转换：

基本NAT-PT机制及原理

在基本NAT-PT机制的工作过程中，当IPv6主机向处于外部公网的IPv4主机发起连接请求时，一组IPv4的地址应该被提前设置好以用来作地址转换使用。从IPv6网络向外传输的IPv6数据包，其源IPv6地址及相关的字段如IP、TCP、UDP和ICMP的头部校验和都将被转换。同理，对于向IPv6网络传输的数据包，也要作类似的转换。采用手工配置或地址池映射，IPv6地址与IPv4地址通过一一对应关系来实现IPv6地址与IPv4地址的转换，可以是静态也可以是动态，提供一对一的IPv6地址和IPv4地址的映射。

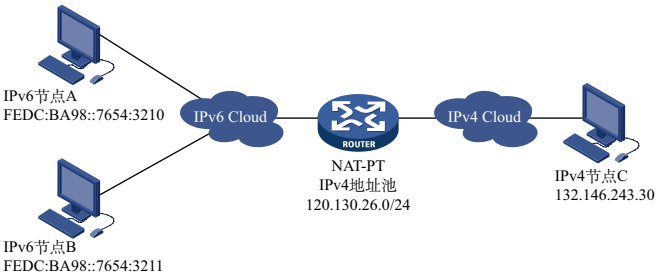


图2 基本NAT-PT原理示意图

如图2所示，NAT-PT设备上配置了一个IPv4的地址池用来做地址转换，地址池中的地址为120.130.26.0/24。地址转换时采用的是静态地址转换，每个IPv6的地址对应一个地址池中IPv4的地址。如果地址池中的地址个数比IPv6网络中的节点数要少，那么NAT-PT地址转换中的动态分配是必须的，否则因为IPv4地址枯竭无法进行地址转换，从而不能建立连接。

假设IPv6网络中的节点A要与IPv4网络中的节点C建立连接进行通讯，节点A创建的数据包信息是：

源地址：SA = FEDC:BA98::7654:3210

目的地址：DA = PREFIX::132.146.243.30

注：前缀PREFIX::/96是由管理员选定的任何可以进行路由的前缀，已经被NAT-PT在IPv6网络中进行了通报，因此以此为前缀的数据包都被路由到NAT-PT设备进行处理。

该数据包到达NAT-PT后，其报文信息将被转换为IPv4版本。如果该数据包不是用来初始化连接的，那么NAT-PT一定保存着该数据包所从属的连接的状态信息，包括IPv6节点A被分配的IPv4地址及其它用于转换的信息。如果该数据包所从属连接的状态信息不存在，或者根本没有这个连接，那么该数据包将被丢弃。如果该数据包是用来初始化连接的，那么NAT-PT将从地址池中分配一个IPv4地址（比如120.130.26.10），并将其报文信息转换为IPv4版本。其传输在整个连接过程中会被存储于内存中，IPv6与IPv4之间的地址映射也将一直被维持着直到连接的结束。

转换后的IPv4报文信息如下：

源地址：SA = 120.130.26.10

目的地址：DA = 132.146.243.30

所有与此连接相关的从节点C返回的IPv4数据包，NAT-PT会根据所存储的连接信息进行信息转换如下：

源地址：SA = PREFIX::132.146.243.30

目的地址：DA = FEDC:BA98::7654:3210

然后，数据包就可以在IPv6网络中进行路由传输了。

NAPT-PT机制及原理

NAPT-PT（Network Address Port Translation - Protocol Translation）把转换的概念做了进一步的扩展，对于传输标识（TCP、UDP的端口号，ICMP标识字段）也进行了转换。不同的IPv6地址转换时，可以对应同一个IPv4地址，通过不同的端口号来区分不同的IPv6主机，从而节省IPv4地址资源。这种机制提供一个IPv4地址和多个使用NAT-PT前缀格式的IPv6地址之间的一对多的动态映射（只有ICMP、TCP和UDP报文可以使用这种方式转换）。

NAPT-PT机制使得多个IPv6节点在与外网IPv4节点进行通讯时可以共用IPv4转换地址。在转换的过程中，IPv6节点的TCP/UDP端口号被转换成已经登记的IPv4的TCP/UDP端口号。传统NAT-PT只局限于TCP、UDP及使用了端口复用的其他应用程序（属于静态地址捆绑），而NAPT-PT解决了传统NAT-PT中存在的问题。例如，当地址池中的地址被耗尽后，NAT-PT将不再起作用，这时，IPv6节点将不能与外部的IPv4节点建立连接。然而，在应用了NAPT-PT后，一个IPv4的转换地址就可以建立63K的TCP和63k的UDP连接。通过修改图2，我们可以使用NAPT-PT进行地址转换，在转换的过程中我们只使用一个IPv4转换地址。

IPv6节点A要与IPv4节点C建立一个TCP连接，创建了如下的数据包地址信息。

源地址：SA = FEDC:BA98::7654:3210，源TCP端口：port = 3017

目的地址：DA = PREFIX::132.146.243.30，目的TCP端口：port = 23

当该数据包到达NAPT-PT后，将分配一个已经被应用到其他连接的IPv4地址用来建立连接，但是其TCP端口号是不同的。于是，上面的数据包地址信息被转换成：

源地址：SA＝120.130.26.10，源TCP端口 port＝1025
目的地址：DA＝132.146.243.30，目的TCP端口 port＝23

当数据从132.146.243.30返回时，根据五元组匹配，数据包被认为是从属于该会话连接，其信息将被转换回IPv6版本如下：
源地址：SA＝PREFIX::132.146.243.30，源TCP端口 port＝23；
目的地址：DA＝FEDC:BA98::7654:3210，目的TCP端口 port＝3017

对于从IPv4网络传向IPv6的连接，每一项服务，NAPT-PT只有一个Server存在，Server的标识是通过TCP/UDP端口号来实现的。例如，IPv6网络中的节点A可能被设置为网络中唯一的HTTP Server（端口号是80）。如果节点C发出如下的数据包信息：
源地址：SA＝132.146.243.30，源TCP端口 port ＝ 1025
目的地址：DA＝120.130.26.10，目的TCP端口 port ＝ 80

那么该数据包将在NAT-PT被转换为：
源地址：SA＝PREFIX::132.146.243.30，源TCP端口 port＝1025
目的地址：DA＝FEDC:BA98::7654:3210，目的TCP端口 port＝80

可以看出目的地址是节点A的IPv6地址。像上面的例子，如果传向NAPT-PT的数据包其目的端口号都是80，那么这些数据包将都被发送给同一个节点，那就是IPv6网络中的节点A。

双向NAT-PT

在双向NAT-PT工作过程中，连接的建立既可以从IPv6侧发起，也可以从IPv4侧发起（而传统的NAT-PT只能从IPv6侧发起）。不管是从IPv6侧还是从IPv4侧发起的连接，也不管其地址的转换机制是静态的还是动态的，经过NAT-PT后，IPv6的地址都要被转换成IPv4地址。 DNS-ALG在双向NAT-PT必须被应用以实现名字向IP地址的转换。IPv6和IPv4两端的DNS，其命名空间是独立且唯一的。特别需要说明的是，当DNS数据包在IPv6和IPv4网络之间传输时，DNS-ALG能够把从IPv6网络中查询到的IPv6地址与地址池中的IPv4地址绑定起来。同样，对于从IPv4网络中查询到的IPv4地址（例如从IPv6端发起对IPv4网络中主机的连接），DNS-ALG同样能够进行类似绑定。

当建立IPv6网络与IPv4网络的连接时，无论是从哪端发起的连接，NAT-PT都将分配一个IPv4转换地址给一个IPv6节点。但是，当初始化一个连接时，从IPv4侧发起连接与从IPv6侧发起连接，其工作原理是不一样的，下面分别进行介绍。

双向NAT-PT：从IPv4到IPv6

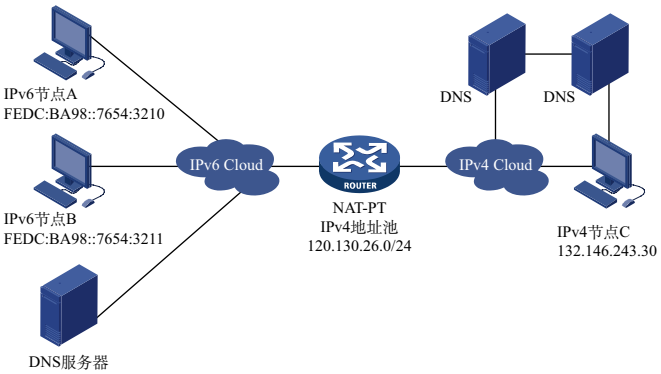


图3 双向NAT-PT原理示意图

如图3所示，当节点C的Name Resolver发出节点A的地址查询请求后，该请求将被传递到IPv6网络中的DNS服务器。因为NAT-PT被配置在IPv6与IPv4网络的交界处，因此查询数据包将会NAT-PT中来回穿梭。在NAT-PT上的DNS-ALG将会对欲进入IPv6网络进行“A”记录查询的请求进行如下修改（NAT-PT根据源或目的端口号是否是53来判断一个数据包是否是DNS数据包）：

- 对于从节点名字到节点地址的查询请求：把查询类型从“A”改成“AAAA”或“A6”；
- 对于从节点地址到节点名字的查询请求：把字符串“IN-ADDR.ARPA”改成“IP6.INT”，把“IN-ADDR.ARPA”前的IPv4地址改成相应的IPv6地址。

反之，当DNS的查询应答欲从IPv6网络进入IPv4网络时，DNS-ALG进行如下操作：

- 更改“AAAA”或“A6”为“A”；
- 把查询到的IPv6地址用NAT-PT分配的IPv4转换地址进行替换。

如果没有提前分配一个IPv4的转换地址给这个IPv6节点，那么现在NAT-PT将分配一个给它。比如上面的例子，节点C要发起与节点A

的连接，产生一个A节点名的查询。这个节点被送到本地的DNS，DNS-ALG截获该报文，并把“A”查询更改为“AAAA”查询或“A6”查询，然后把它发送给IPv6网络的DNS服务器。DNS服务器将做出如下应答：

Node-A　　AAAA　　FEDC:BA98::7654:3210

该应答仍然会被DNS-ALG捕获，并转换成如下IPv4形式：

Node-A　　A　　120.130.26.1

在NAT-PT上，DNS-ALG将会维持FEDC:BA98::7654:3210和120.130.26.1的映射关系。现在这条“A”记录就可以返回给节点C了，从而节点C将发起如下的连接信息：

源地址：SA = 132.146.243.30，源TCP端口 port = 1025

目的地址：DA = 120.130.26.1，目的TCP端口 port = 80

带有此信息的数据包将被路由到NAT-PT，因为它维护着FEDC:BA98::7654:3210和120.130.26.1的映射关系，因此上述信息将被转换成：

源地址：SA = PREFIX::132.146.243.30，源TCP端口 port = 1025

目的地址：DA = FEDC:BA98::7654:3210，目的TCP端口 port = 80

连接由此建立了起来，通讯就可以进行了。

对于上面所描述的从IPv4到IPv6的连接建立过程，可能会导致服务攻击拒绝（denial of service attack）。因为一个节点可以发起多个查询，导致NAT-PT把IPv4的地址资源耗尽，从而阻止了后面的服务。因此对于此种连接的建立（从IPv4向IPv6发起的连接）应该有一个超时时间以降低服务拒绝的可能性。在从IPv6到IPv4的连接建立过程中，可以提前保存一个IPv4的地址（采取NAPT-PT）来减少在建立过程中服务拒绝的可能性。

双向NAT-PT：从IPv6到IPv4

IPv6节点通过IPv4网络中的或者是IPv6网络中的DNS服务器来获取IPv4节点的IP地址。建议IPv6网络上的DNS服务器能够维持IPv6节点名字与地址的映射关系，并能够缓存IPv6地址与外部IPv4地址的对应关系。如果IPv6网络中的DNS服务器包含与外部IPv4网络节点的地址映射，那么DNS查询就不会穿过IPv6网络，也不需要DNS-ALG的干预。否则，查询就要穿过IPv6网络，并需要DNS-ALG的协同工作。建议IPv4网络中的DNS Server只映射IPv4节点与IPv4地址之间的映射关系，不推荐在其上进行跨IPv4网络和IPv6网络的转换。

对于从IPv6到IPv4的连接，在NAPT-PT模式下，一旦新的连接要建立，TCP/IP的源端口号将从已登记的IPv4地址进行分配。另外，在连接建立过程中，地址前缀（PREFIX :/96）的使用，不会影响IPv4节点上的任何配置。

现在仍然以图3为例，假设节点A要与节点C建立一个连接，因此节点A要发起对节点C的名字查询（“AAAA”或“A6”）。因为节点C可能有IPv6地址，也可能有IPv4地址，NAT-PT上的DNS-ALG就把这个原始的AAAA/A6查询直接转发给外部的DNS系统，同时还发送了一个A查询。如果目的节点（节点C）的AAAA/A6记录存在，那么，信息将会被传回到NAT-PT，然后NAT-PT会直接把信息再传回到节点A。相反，如果节点C的A记录存在，那么，记录信息也会被传给NAT-PT，DNS-ALG就会添加上合适的前缀并把信息转发给发起查询的设备，即节点A。所以，如果A记录如下：

节点C　　A　　132.146.243.30，

那么就会被转换成

节点C　　AAAA　　PREFIX::132.146.243.30　　或者

节点C　　A6　　PREFIX::132.146.243.30

现在节点A就可以使用这个地址建立连接，就像它与其他IPv6的节点进行通讯一样。

有一个问题值得注意：那就是IPv6网络中的DNS服务器是如何与IPv4网络进行对话的，或者反过来，IPv4网络中的DNS服务器如何与IPv6网络进行对话，要知道在这儿没有双栈的节点。具体的实现是这样的，IPv6网络中的DNS服务器有一个映射的IPv4地址，该地址可能是NAT-PT地址池中的一个，对NAT-PT来说是可知的。NAT-PT维持着该IPv4地址与IPv6网络中的DNS服务器之间的一一映射关系。而另一个方向，IPv4网络中的DNS服务器有一个映射的IPv6地址，该IPv6地址由外部的IPv4 DNS服务器与前缀（PREFIX::/96）构成。NAT-PT同样维持着该IPv6地址与IPv4网络中的DNS服务器之间的一一映射关系。通过这样的映射，实现了不同类型网络中DNS服务器与主机的互通。

NAT-PT转换细节

SIIT-无状态IP/ICMP转换

SIIT（Stateless IP/ICMP Translation Algorithm）是NAT-PT的基础，指无状态IP/ICMP转换算法，在RFC2765中有详细描述，它解决了IPv6与IPv4节点之间进行通讯的问题。IPv6节点通过获取一个暂时的IPv4地址，为自己配置一个IPv4翻译地址，而作为IPv4网络侧的IPv4节点通过IPv4映射地址与IPv6节点之间进行通讯。在通讯的过程中，转换器将对报文头进行转换，使得报文在IPv6网络和IPv4网络中都能够被顺利的路由并建立和维持通讯。

IPv4向IPv6报文头转换

当转换器收到从IPv4网络到IPv6网络的报文时候，就要把IPv4的报文头转换为IPv6的报文头，原先的IPv4报文头就被移走，用新的IPv6的报文头代替。除了ICMP包以外，传输协议报头和数据部分都保留不变。

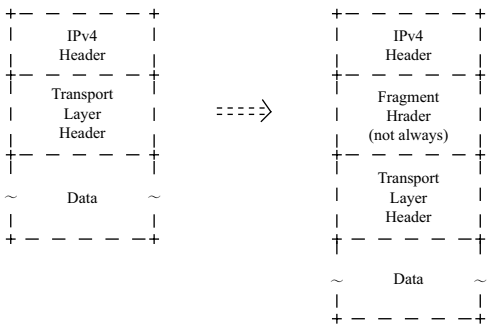


图4 IPv4向IPv6报文转换

另外，由于IPv6协议和IPv4协议在处理报文分片方式上的不同，导致在处理分片报文时需要特别注意。对于IPv6来说，只有发送端可以对报文进行分段，而中间路由器不能进行分段。因此对于IPv6，path MTU Discovery就是必须的了，而对于IPv4来说，中间路由器可以进行分段，所以path MTU Discovery也就是可以选择的了。所以，如果IPv4的报文头中DF位没有设置，那么为了保证转换后的IPv6报文不会超过MTU，则需要数据净荷长度不能超过1232字节（最小MTU1280字节–40字节IPv6报头–8字节分段报头），如果DF位被设置了，同时报文不是一个分段报文的话，则转换后的IPv6报文不需要包含分片报头，此时IPv6报文头各个字段设置如下：

IPv6字段	设置
Version（版本）	6
Traffic Class（业务流类别）	直接从IPv4报头的Type Of Service and Precedence域中所有8字节copy过来
Flow Label（流标签）	0
Payload Length（净荷长度）	IPv4报头中Total length值减去IPv4报头的长度和IPv4选项的大小
Next Header（下一报头）	从IPv4报头中的Protocol域中copy过来
Hop Limit（跳极限）	从IPv4报头中的TTL域中copy过来
Source Address（源IP地址）	低32位是IPv4的源地址，高96位是IPv4-mapped前缀（::ffff:0:0/96）
Destination Address（目的IP地址）	低32位是IPv4的目的地址，高96位是IPv4-translated前缀（0::ffff:0:0:0/96）

表1 IPv4向IPv6报头转换字段列表（无分片）

如果转换后的IPv6报文需要添加分片报文头（原始IPv4报文DF位没有被设置，或者本身就是一个分片），那么报文需要做如下修改：

IPv6字段	设置
Payload Length（载荷长度）	在原来的IPv4基础上加上分段头长度8（IPv4报头中length域的值减去IPv4报头和选项的大小，再加上8字节的分段报头长度）
Next Header（下一报头）	分片头（44）
分片报头字段	
Next Header（下一报头）	从IPv4报头的Protocol域中copy过来
Fragment Offset（分片偏移量）	从IPv4的Fragment Offset中copy过来
M flag（M标记位）	从IPv4报头的More Fragments域中copy过来
Identification（分片标识）	低16位从IPv4报头的Identification域中copy过来，高16位为0

表2 IPv4向IPv6报头转换列表（有分片）

ICMPv4向ICMPv6报文头转换

所有的ICMPv4报文向IPv6报文的转换都要重新计算其校验和，因为ICMPv6报文和UDP/TCP一样有一个伪报头的校验和。另外所有类型（Type）字段都需要更新。如果是ICMP错误报文，那么错误信息中的IP头也需要转换。最后，前面的调整也可能对报文长度产生影响，因此还需要对相应的域进行修改。

	IPv4中Type值	转换处理
Echo and Echo Reply（请求回显和回显应答）	Type 8 and 0	type to 128 and 129
Information Request/Reply（信息请求和信息应答）	Type 15 and 16	Silently drop
Timestamp and Timestamp Reply（时间戳请求和时间戳应答）	Type 13 and 14	Silently drop
Address Mask Request/Reply（地址掩码请求和应答）	Type 17 and 18	Silently drop
ICMP Router Advertisement（路由器通告）	Type 9	Silently drop
ICMP Router Solicitation（路由器请求）	Type 10	Silently drop
Unknown ICMPv4 types		Silently drop

表3 ICMPv4向ICMPv6查询报文转换列表

对于ICMPv4差错报文的转换（下表没有明确标出Type的，转换后Type值为1）：

Destination Unreachable（Type 3）目的不可达	Code值	转换处理或者对应ICMPv6中code值
net，host unreachable（网络、主机不可达）	Code 0，1	0（没有可达路由）
protocol unreachable（协议不可达）	Code 2	ICMPv6参数问题（Type 4，Code 1），指针指向IPv6的下一个扩展头字段
port unreachable（端口不可达）	Code 3	4（端口不可达）
fragmentation needed and DF set（需要分片但DF为置位）	Code 4	ICMPv6包过大（Type 2，code 0），如果包中的MTU域没有填充，则转换器需要自己去确定一个可能的path MTU来填充此域
source route failed（源站选路失败）	Code 5	0（没有可达路由）
	Code 6，7，8	0（没有可达路由）
communication with destination host administratively prohibited（目的网络、主机被强制禁止）	Code 9，10	1（与非法主机通讯）
	Code 11，12	0（没有可达路由）
Redirect（重定向）	Type 5	Silently drop
Source Quench（源端被关闭）	Type 4	Silently drop
Time Exceeded（超时）	Type 11	Type 3，code部分不变
Parameter Problem（参数问题）	Type 12	Type 4，相应域需要进行改变

表4 ICMPv4向ICMPv6差错报文转换列表

IPv6向IPv4报文头转换

当一个转换器收到一个IPv4映射地址的报文时，那么该IPv6报文将被转换成IPv4报文，然后基于该IPv4目的地址进行转发，从而将IPv6报文头转换成IPv4报文头。除了ICMP报文，其他的传输层报文头和数据部分不会发生任何变化。

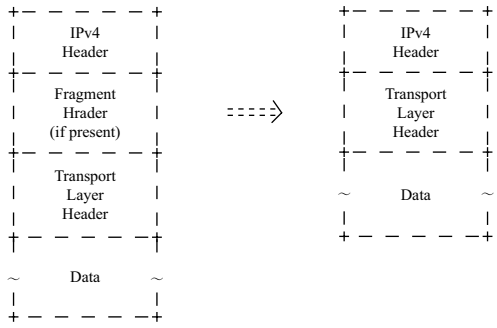


图5 IPv6向IPv4报文转换

如果没有IPv6分片头，那么转换后的IPv4报文头的字段设置如下表：

IPv4字段	设置
Version（版本）	4
Internet Header Length（首部长度）	5（no IPv4 options）
Type of Service and Precedence（服务类型）	从IPv6的Traffic Class域中copy过来
Total Length（总长度）	IPv6报头中Payload length域的值加上IPv4报头大小
Identification（标识）	0
Flags（标识位）	MF域置为0，DF域置为1
Fragment Offset（分片偏移）	0
Time to Live（生存时间）	从IPv6的Hop Limit域中copy过来
Protocol（协议）	从IPv6中的Next Header域中copy过来
Header Checksum（首部检验和）	IPv4报头创建好重新计算
Source Address（源地址）	如果IPv6源地址是IPv4-translated地址，则IPv6地址的低32位copy到IPv4源地址域。其他的情况源地址都设为0.0.0.0
Destination Address（目的地址）	IPv4-mapped地址的低32位

表5 IPv6向IPv4报头转换字段列表（无分片）

如果IPv6报文中含有分片头，那么除了下面的特殊字段变化不同以外，其他的仍然根据上面的叙述进行转换：

IPv4字段	设置
Total Length	IPv6报头中的Payload length域的值加上IPv4报头的大小，减去8（Fragment header的大小）
Identification	Copy IPv6中的分片头中的Identification域的低16位
Flags	MF值引用分片头中的M标志，DF设置成0，标识运行IPv4路由器进行分片
Fragment Offset	从分片头中的分片位移域copy过来
Protocol	从分片头中Next Header域中copy过来

表6 IPv4向IPv6报头转换字段列表（有分片）

ICMPv6向ICMPv4报文头转换

因为ICMPv6同TCP、UDP一样具有伪包头校验和，因此所有需要转换的ICMPv6的报文，其校验和字段必须被更新。另外ICMP的类型字段及包含IP报文头的ICMPv6错误报文的内容也需要同步更新。

	IPv6中Type值	转换处理
Echo Request and Echo Reply（请求回显和回显应答）	128 和129	type to 0 and 8
MLD Multicast Listener Query/Report/Done（MLD 多播侦听查询/汇报/完成）	Type 130，131，132	Silently drop
Neighbor Discover messages（邻居发现信息）	Type 133 through 137	Silently drop
Unknown informational messages（未知信息类型）		Silently drop

表7 ICMPv6向ICMPv4查询报文转换列表

差错报文目的不可达的Type值为1，转换后Type为3，转换code域方法如下：

	Code值	转换处理或IPv4中对应的Code值
no route to destination（没有到达的路由）	Code 0	Code 1（host unreachable）（主机不可达）
communication with destination administratively prohibited（与管理拒绝的目的主机的通讯）	Code 1	Code 10
beyond scope of source address（超出源地址范围）	Code 2	Code 1（host unreachable）
address unreachable（地址不可达）	Code 3	Code 1（host unreachable）
port unreachable（端口不可达）	Code 4	Code 3（port unreachable）

表8 ICMPv6向ICMPv4差错报文转换列表

	Type值	转换处理或对应IPv4中Type值
Packet Too Big（包过大）	Type 2	转换为ICMPv4中目的不可达（code 4），MTU域需要对报文是否有分段而进行更新
Time Exceeded（超时）	Type 3	Type为11，Code不变
Code field is unchanged（Code问题）	Type 4	如果Code值是1转换为协议不可达（Type 3，Code 2）；其他情况Type为12，code为0，指针需要针对相应的域进行转换包括IP头
Unknown error messages（未知错误报文）		Silently drop

表9 ICMPv6向ICMPv4其他报文转换列表

NAT-PT转换细节

为了使IPv6与IPv4网络之间的端对端交流成为可能，NAT-PT需要把IPv6和IPv4的报文头进行转换。由于地址转换功能和端口复用的使用，NAT-PT还要对上层协议报文头做相应的调整。协议转换的详细介绍在SIIT中，但是因为NAT-PT也进行了地址转换，因此在NAT-PT上还有一些补充的修改。

IPv4 报头向IPv6报头的转换

在NAT-PT的具体应用中，除了下面的字段说明之外，其他字段的转换的过程和RFC2765（SIIT）中所描述几乎是一样的。

IPv6字段	设置
源地址	低32位是IPv4的32位源地址，高96位是被指定的前缀（PREFIX::/96）符合此类地址格式的数据包将被路由到NAT-PT网关
目的地址	NAT-PT保持着IPv4目的地址和IPv6目的地址之间的映射关系，根据这种映射关系IPv4目的地址被IPv6地址所替代

表10 NAT-PT中IPv4向IPv6报头转换

IPv6报头向IPv4 报头的转换

在NAT-PT的具体应用中，除了下面所描述的源地址说明之外，其他的转换方式和RFC2765（SIIT）中所描述的一样。

IPv4字段	设置
源地址	NAT-PT保持着地址池中的IPv4地址与IPv6源地址之间的映射关系，根据这种映射，IPv6源地址将被IPv4地址所替代
目的地址	被转换的IPv6数据包的目的地址将遵循如下格式：PREFIX ::IPv4/96，因此IPv6目的地址的低32位将被直接复制为IPv4的目的地址

表11 NAT-PT中IPv6向IPv4报头转换

从IPv4到IPv6校验和的更新

UDP的校验和（当非零时）和TCP的校验和应该被重新计算，从而反映从IPv4到IPv6的地址变化。校验和的计算法则可以借鉴RFC1631（NAT）。在NAPT-PT的转换模式下，TCP/UDP校验和的变化除了包括地址的变化外，还有包括TCP/UDP端口号的变化。

如果发送到NAT-PT的IPv4 UDP数据包的校验和为0，NAT-PT在把它转换成IPv6数据包后要重新对该UDP报文做校验和计算。如果校验和为0的UDP报文是是以分片的形式发送过来的，那么NAT-PT需要做的是等待所有的报文到达，重新聚合成一个完整的报文后，进行IPv6的转换，然后进行校验和计算，最后再转发出去。

IPv6中的ICMP报文不同于IPv4中的ICMP报文，它采用了伪包头，对于校验和的计算类似于UDP和TCP的计算过程（此部分可参考RFC2460的8.1节）。需要注意的是，由于ICMP报文的载荷中也可能携带有地址信息，该部分同样要做地址转换，转换后才能够进行校验和的计算。

从IPv6到IPv4校验和的更新

同从IPv4到IPv6的TCP/UDP/ICMP校验和一样，它的变化反映了从IPv6到IPv4的地址变化，计算法则也是一样的。在NAPT-PT的转换模式下，此变化还包括端口号的改变。对于UDP报文，因为其校验和是可选项，所以在进行转换后，校验和可以被简单的设置成0。而ICMP报文的校验和必须要重新计算，因为IPv4中无ICMP伪报头的概念（同样需要注意ICMP载荷中携带地址信息的情况）。

NAT-PT的局限

在RFC2663中描述的关于NAT的局限性同时也就是NAT-PT的局限性，下面所列举的是局限性中其中比较重要的，也有些是NAT-PT所独有的局限性。

拓扑局限性

使用NAT-PT是受局限的，所有与某进程有关的请求与应答都必须要经过同一NAT-PT路由器。实现这一点的一个方法就是把NAT-PT置于边界路由器上，该路由器对于一个局域网络来说是唯一的，这样所有的IP数据包或者是从该网络发起的，或者是以该网络为目的的。对于NAT来说，这同样是个问题，并在RFC2663中进行了描述。

协议转换限制

一些IPv4的数据报字段在IPv6中发生了改变，因此NAT-PT的转换是不很准确的。例如IPv4中的选项段在IPv6中语义上有很大的改变，就不能很好的进行转换。

需要ALG的协调工作

因为NAT-PT需要对地址进行转换，那么当某个上层的应用程序，在其应用中包含了地址信息（或端口信息），如果NAT-PT不进行特殊处理，就会引起使用问题。在这种情况下，应用层网关（ALG：Application Layer Gateway）就需要为这类应用提供协调服务。这对于IPv4的NAT来说也是普遍的问题。

缺少端对端的安全性

NAT-PT最大的局限性之一就是在端对端网络层上缺少安全性。另外，对于传输层及应用中携带IP地址的应用层也没有安全性可言。这是NAT的一个遗传问题。且不说NAT-PT，对于跨两个不同IP地址的网络来说，IPsec是不可能实现的。两个端点之间要实现网络层上的IPsec功能，必须都是IPv4节点或者都是IPv6节点。

DNS转换和DNS-SEC

前面给出了DNS信息转换的机制，但是很明显这种机制不能和安全DNS同时使用。例如，IPv6网络中的授权DNS服务器收到来自IPv4网络的查询时，它无法对回复进行签名，从而导致域名解析失败。

H3C防火墙NAT“无限次”转换说明

文/王军

NAT（Network Address Translation，网络地址转换）是将IP数据包报头中的IP地址转换为另一个IP地址的过程。最初，IP地址本身并没有私网和公网之分，只是我们人为的定义了一部分IP地址作为私网地址使用，区别于公网地址。NAT的使用场景是当私网用户访问Internet的报文经过NAT设备时，NAT设备会用一个合法的公网地址替换原报文中的私网源IP地址，并对这种转换进行记录；之后，当报文从Internet侧返回时，NAT设备查找原有的记录，将报文的目的地址再替换回原来的私网地址，并回送给发出请求的主机。这样，在私网侧或公网侧设备看来，这个过程与普通的网络访问并没有任何的区别，也并不感知NAT设备的存在。

NAT是一种私网地址与公网地址之间的一种转换，那么NAT设备就需要准备一定数量的公网地址，公网地址数的多少一方面取决于内网用户的多少，另一方面也取决于NAT设备的转换算法。如何最大化的利用IPv4地址资源，节约IPv4地址数量，本文就H3C防火墙的NAT无限次连接进行了说明。

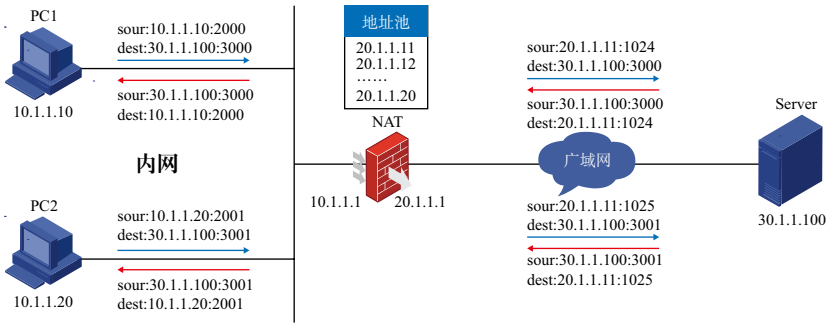
有限次NAT转换

在阐述NAT无限次转换之前，我们先了解一下，一般的NAT转换对IPv4地址的需求。

参考本册大爬虫的《NAT概述》，我们知道NAT按照端口映射方式分类，可以分为如下4种：全锥形NAT、限制锥形NAT、端口限制锥形NAT和对称型NAT。前三种NAT的模型都是锥形（Cone）NAT，有一个共同点：只要是从同一个内部地址和端口（LocalIP:LocalPort）出来的包，NAT都将它转换成同一个外部地址和端口（PublicIP:PublicPort），而不同的内部地址和端口（LocalIP:LocalPort）出来的包，都转换成不同的外部地址和端口（PublicIP:PublicPort），即转换后的外部地址和端口不能复用，导致NAT转换次数受限。

不同于锥形（Cone）NAT，对称（Symmetric）NAT的主要特点为从同一个内部地址和端口组合（LocalIP:LocalPort）发出的包，NAT可以将它转换成不同的外部地址和端口组合（PublicIP:PublicPort）。这样一来，一个内部地址和端口组合（LocalIP:LocalPort）可以对应多个公网地址和端口组合（PublicIP:PublicPort），如何标记一个（LocalIP:LocalPort）——多个（PublicIP:PublicPort）的对应关系呢？如果每个转换后的公网地址和端口组合（PublicIP:PublicPort）必须不同，则NAT转换数量仍然是有限的。

如下图所示，若地址池中有10个IP地址，内网主机访问外网服务器，每次访问经过防火墙做NAT，分配的（IP:Port）必须各不相同。此时，防火墙最多可以NAT转换的次数为地址池IP地址数乘以可以转换的端口数：10*（65536-1024）。



NAT转换原理图

H3C防火墙“无限次”NAT转换

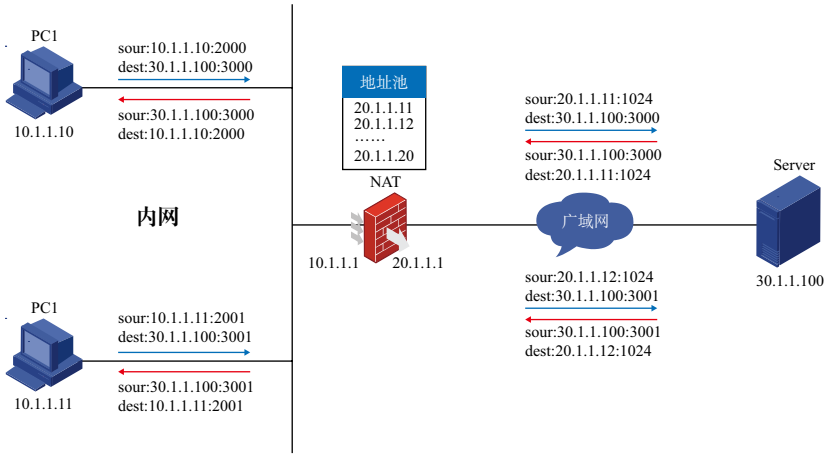
一般的设备用（LocalIP:LocalPort）——（PublicIP:PublicPort）标识一次NAT转换，只能实现有限次的NAT转换。怎样才能实现无限次的NAT转换呢？H3C防火墙将NAT模块与会话管理模块相结合。每一个会话都有五元组，如果在（LocalIP:LocalPort）——（PublicIP:PublicPort）的基础上再加上会话信息一起来记录一次NAT转换，那么每次转换记录的信息将变为（LocalIP:LocalPort）——（PublicIP:PublicPort）——（Dest-IP:Dest-Port）及其协议类型，并用转换后的会话五元组唯一标识一次NAT转换。而公网中的（Dest-IP:Dest-Port）可以认为是无限多的，这时防火墙NAT转换的IP＋端口（PublicIP:PublicPort）资源便可以根据（Dest-IP:Dest-Port）的不同而复用，所以防火墙的NAT转换次数也将是无限多的（公网中所有的（Dest-IP:Dest-Port）即表示了公网中所有基于IP的应用，从这个意义上说H3C防火墙的NAT转换是无限多次的）。

出方向的NAT主要包括静态NAT、Basic NAT（No-PAT）、NAPT、Easy IP。其中静态NAT是一对一的IP地址转换，转换次数与配置的地址数有关，是有限次的；Basic NAT不涉及端口的转换，其NAT转换后防火墙记录的会话数可以无限多次，但是防火墙NAT模块工作的部分只是IP地址的转换，端口并不需要防火墙处理，所以防火墙Basic NAT能够转换的最大次数只根据地址池的大小而定，也是有限次的。本文重点讨论H3C防火墙NAT无限次转换的原理，即NAPT和Easy IP方式的NAT转换，其因为转换后可以复用（PublicIP:PublicPort）从而实现了NAT转换次数的无限次。

NAPT方式原理

NAPT方式属于多对一的地址转换，通过使用“IP地址＋端口号”的形式进行转换，使多个私网用户可共用一个公网IP地址访问外网。因此是地址转换实现的主要形式。

防火墙NAPT地址转换原理如下：



NAPT方式原理图

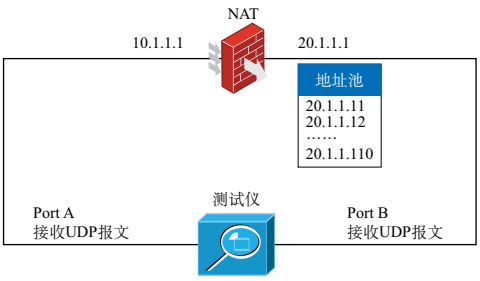
如上图所示，防火墙NAPT方式的处理过程如下：

■ 防火墙收到私网侧主机发送的访问公网侧服务器的报文，按照五元组建立会话表项（正反向）；

■ 防火墙从地址池中按照一定的规则选取一对空闲的“公网IP地址＋端口号”，建立与私网侧报文“源IP地址＋源端口号”间的NAPT转换关系（正反向），与（1）中建立的会话表项相结合，将转换后的地址＋端口填充到会话表项中，并依据查找正向会话表项的结果将报文转换后向公网侧发送；

■ 防火墙收到公网侧的回应报文后，根据其五元组查找反向会话表项，并依据查表结果将报文转换后向私网侧发送。

下面我们通过三个实验来验证H3C防火墙NAPT的地址转换方式、无限次转换说明和无限次转换的应用限制。实验设计：测试仪Port A向Port B发送UDP报文，防火墙做NAPT，地址池中有100个IP地址20.1.1.11—20.1.1.110。实验基本组网图如下：



NAPT实验组网图

H3C防火墙NAPT地址转换IP、端口分配说明

H3C防火墙NAPT方式转换地址、端口分配规则如下：

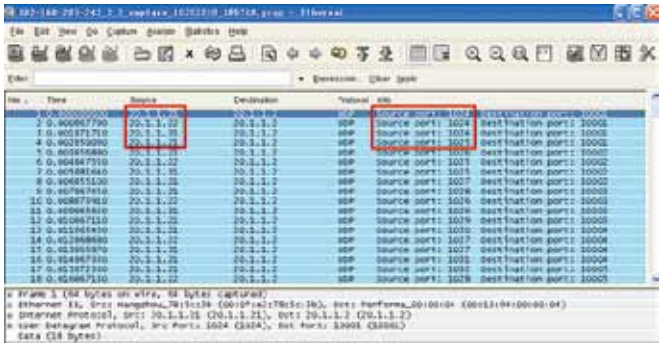
■ 地址分配：根据地址池中的地址个数，系统自动生成N个数值，每一个数值对应标识一个地址池中的IP地址。匹配上ACL的数据报文，根据报文的源IP地址，由hash算法算出一个值，其值一定会在系统自动生成的N个值之内产生，由这个值来决定用哪个IP地址；

■ 端口分配：确定使用地址池中某个IP地址后，该IP对应的端口号按照从1024-65535之间从小到大，以1为步长分配。如果是两个不同源IP地址的数据报文匹配上ACL，而根据地址分配规则正好分配到同一个IP，其端口分配根据收到报文的先后顺序，第一个源IP的报文分配端口M，第二个源IP的报文分配端口M＋1。

实验一：测试仪打流设置：测试仪A口使用4个IP地址10.1.1.10、10.1.1.11、10.1.1.20、10.1.1.110为源地址，源端口固定，目的IP固定，目的端口离散10000次。我们通过在测试仪接收端口抓包来分析防火墙NAPT地址转换IP、端口分配。

结果：测试仪Port A打入的报文经过NAT转换后IP都为20.1.1.21、20.1.1.22、20.1.1.31，因为测试仪是按照轮询顺序打入四个源IP的报文，通过抓包可以发现源地址为10.1.1.10、10.1.1.110的报文转换后源地址都为20.1.1.21，同时10.1.1.11、10.1.1.20转换后源地址分别为20.1.1.22、20.1.1.31。说明防火墙在处理源地址时，两个私网源地址10.1.1.10、10.1.1.110 hash得到同一个值，所以转换后分配到同一个IP。此外，转换后的源端口根据每个转换后源IP地址各自单独分配，从1024开始，以1为步长增加。即20.1.1.21、20.1.1.22、20.1.1.31三个地

址对应的端口都单独从1024开始分配，以1为步长增加，不管转换时（LocalIP:LocalPort）——（PublicIP:PublicPort）的对应关系中有多少个源地址。



实验一 测试仪接收端口抓包

H3C防火墙NAPT无限次转换说明

实验二：测试仪打流设置：以10.1.1.10、10.1.1.110两个不同的IP为源地址，由实验一知，防火墙NAPT转换后分配的IP地址相同。此时，如果NAPT转换后的（PublicIP:PublicPort）组合不能复用，地址池中一个IP对应只有64512个端口可以用于转换，则一个地址只能转换64512次；如果可以复用，就可以转换无限多次。现设置源地址为10.1.1.10、10.1.1.110的流，源IP10.1.1.10的源端口离散50000次（从10001~60000），目的IP唯一为20.1.1.2，目的端口唯一为2001。源IP10.1.1.110的源端口离散50000次（从10001~60000），目的IP唯一为20.1.1.2，目的端口唯一为2002。

结果：经过NAPT转换后，两个私网源地址经过NAT转换后的源IP都为同1个IP地址20.1.1.21，防火墙可以建立100000条会话，转换次数大于64512次，即说明转换后的IP＋端口组合（PublicIP:PublicPort）可以根据目的IP＋端口复用；转换后源端口按照端口分配规则，从1024开始分配，以1为步长增加。从实验二测试仪接收端口抓包中用UDP端口条件udp.port＝3098进行筛选，可以看出，有2次NAPT转换都是使用相同公网IP＋源端口（20.1.1.2＋1024），说明如果某一个公网IP的端口号耗尽，只要（Dest-IP:Dest-Port）不同，转换后的源端口是会重新开始使用1024端口的，而且可以正常建立会话表项，即复用转换后的IP＋端口组合（PublicIP:PublicPort），实现无限次转换。

```
<H3C>display session statistics
Current session(s):100000
        Current      TCP session(s): 0
                Half-Open: 0              Half-Close: 0
        Current      UDP session(s): 100000
        Current     ICMP session(s): 0
        Current   RAWIP session(s): 0

Current relation table(s): 0

Session establishment rate:          0/s
        TCP     Session establishment rate:          0/s
        UDP     Session establishment rate:          0/s
        ICMP    Session establishment rate:          0/s
        RAWIP   Session establishment rate:          0/s

Received      TCP:                    0 packet(s)                    0 byte(s)
Received      UDP:               100004 packet(s)              8200312 byte(s)
Received     ICMP:                    0 packet(s)                    0 byte(s)
Received   RAWIP:                      0 packet(s)                    0 byte(s)
Dropped       TCP:                     0 packet(s)                    0 byte(s)
Dropped       UDP:                     0 packet(s)                    0 byte(s)
Dropped      ICMP:                     0 packet(s)                    0 byte(s)
Dropped    RAWIP:                      0 packet(s)                    0 byte(s)
<H3C>
```

实验二 防火墙会话数100000条

实验二 测试仪接收端口抓包

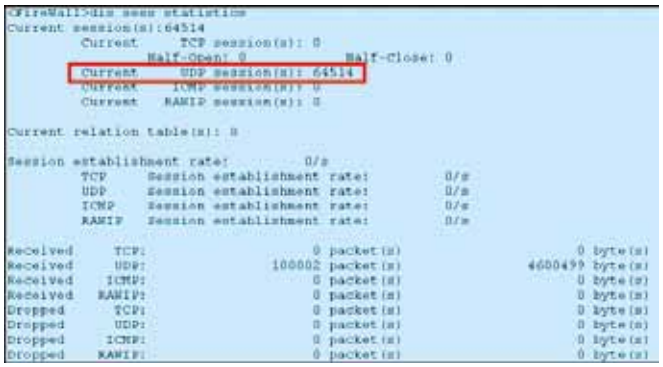
实验二 转换后（PublicIP:PublicPort）复用会话

H3C防火墙NAPT无限次转换的应用限制

无限次转换适用于是在NAT转换后五元组不重复的场合，有一种特殊的场景，当内网中存在大量的用户访问同一台服务器的同一服务（即同一端口）时，NAPT转换后的（PublicIP:PublicPort）组合不能复用，地址池中一个IP对应只有64512个端口可以用于转换，则一个地址只能转换64512次。因为当公网地址池中的一个IP已经转换了64512次访问同一个（Dest-IP:Dest-Port），再增第64513次访问时，该IP就不能再分配空余的端口，假设再次分配1024端口，则和就会存在两次（PublicIP:PublicPort）——（Dest-IP:Dest-Port）的映射关系相同，返回的报文无法识别发往哪个内网用户，NAT转换连接数受限。所以H3C防火墙的NAT转换仍然存在转换数限制。

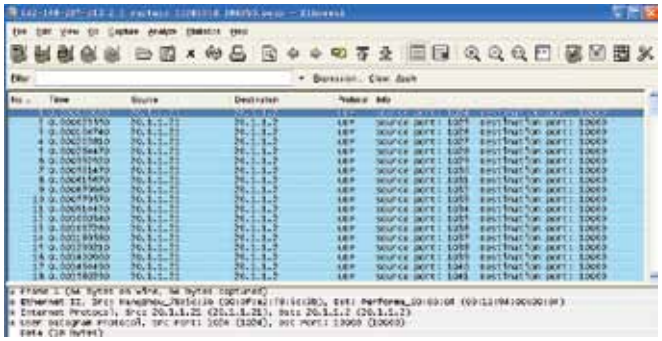
实验三：测试仪打流设置：以10.1.1.10、10.1.1.110两个不同的IP为源地址，由实验一可知，防火墙NAPT转换后分配的IP地址相同。现设置源地址为10.1.1.10、10.1.1.110的流，源端口都离散50000次（从10001~60000），目的地址都为20.1.1.2，目的端口固定为10000。流量设置测试仪brust方式发送100000个报文。

结果：经过NAPT转换后，转换后的源IP为1个IP地址20.1.1.21，查看防火墙会话总数，可以建立64514条会话（防火墙有其他local域等会话产生多余计数，查看测试仪抓包计数可知由于打流产生的会话数量应是64512），无法建立100000条有效连接，即说明由于目的IP＋端口组合（Dest-IP:Dest-Port） 固定，导致（PublicIP:PublicPort）无法复用，最大连接数量受限；前64512个报文可以正常NAT转换，转换后源端口按照端口分配规则，从1024开始分配，以1为步长增加。第64513个报文开始，NAT转换不成功，无法建立新的连接。



实验三 防火墙会话

实验三 测试仪接收端口收包数为64512



实验三 测试仪接收端口抓包

EASY IP方式原理

EASY IP方式是指直接使用接口的公网IP地址作为转换后的源地址进行地址转换，它可以动态获取出接口地址，从而有效支持出接口通过拨号或DHCP方式获取公网IP地址的应用场景。同时，EASY IP方式也可以利用访问控制列表来控制哪些内部地址可以进行地址转换。

H3C防火墙EASY IP方式NAT转换的地址分配方式为固定为接口IP地址。端口分配方式与NAPT相同，虽然转换后的IP地址只有一个，可以做到根据目的IP＋端口的不同复用转换后的IP＋端口组合（PublicIP:PublicPort），所以可以实现无限次转换。可以认为EASY IP方式除了转换后IP地址固定为接口地址以外，其他处理方式基本与NAPT方式相同，在此不再详述。

小结

当然了，任何产品的表项规格，都会受限于产品的内存或芯片容量等因素，H3C的防火墙也不例外，本文主要描述原理上的无限次NAT转换连接。

H3C防火墙理论上支持NAT无限次转换，加上本身强大的会话处理能力，可以用极少的公网IP地址作为整个局域网的出口网关，可以很大程度上节约了公网IP资源，延缓IPv4地址的枯竭。

《网络大爬虫》

Network Addicts

《网络大爬虫》是H3C面向H³Care俱乐部VIP会员技术爱好者的专业性技术刊物。本刊主要深入探讨IP相关技术，内容涉及交换、OSPF路由协议、QoS、MPLS VPN等多个技术领域。本刊的文章由H3C相关领域技术专家倾力撰文。如有建议或需求，请您反馈至电子信箱：H³Care_club@h3c.com，感谢您的阅读！

华三服务以满足客户业务需求为导向，依托广博精深的技术实力、完备高效的交付体系，贴近客户，持续进行服务创新，为客户提供温暖感、专业化的服务体验，帮助客户实现 IT 价值的可持续性再生。

H3care

温暖感 专业化

客户服务热线

400-810-0504
800-810-0504

杭州华三通信技术有限公司

杭州基地
杭州市高新技术产业开发区之江科技
工业园六和路310号
邮编：310053
电话：0571 – 86760000
传真：0571 – 86760001

北京分部
北京市宣武门外大街10号庄胜广场中
央办公楼南翼16层
邮编：100052
电话：010–63108666
传真：010–63108777

http://www.h3c.com.cn

CN-125X30-20120521-BR-LR-V1.0