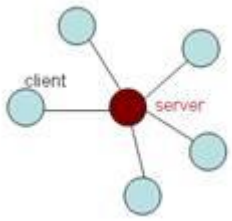
P2P 中的 NAT 穿越方案

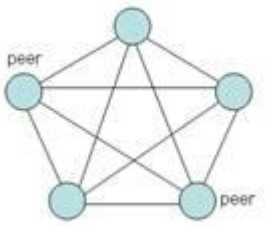
文/王军

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（如左图所示）有着明显的区别。P2P 可以是一种通信模式、一种逻辑网络模型、一种技术、甚至一种理念。在 P2P 网络中（如右图所示），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。



CS 模式



P2P 结构模型

NAT 技术和 P2P 技术作为时下最热门的两项网络技术，在现在的网络上有着广泛的应用，P2P 主机位于 NAT 网关后面的情况屡见不鲜。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，但是 P2P 技术却要求通信双方都能主动发起访问，所以要在 NAT 网络环境中进行有效的 P2P 通信，就必须采用新的解决方案。

P2P 作为一项新兴技术，有很大的优化空间，并且相对于网络设备，基于 P2P 的应用程序在实现上更为灵活。所以为了兼容 NAT，基于 P2P 的应用程序在开发的时候大多会根据自身特点加入一些穿越 NAT 的功能以解决上述问题。以下着重介绍几种常见的 P2P 穿越 NAT 方案。

一、 反向链接技术

适用场景：P2P 通信双方中只有一方位于 NAT 设备之后

如图 1 所示，客户端 A 位于 NAT 之后，它通过 TCP 端口 1234 连接到服务器的 TCP 端口 1235 上，NAT 设备为这个连接重新分配了 TCP 端口 62000。客户端 B 也通过 TCP 端口 1234 连接到服务器端口 1235 上。A 和 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 B 拥有外网 IP 地址，所以 A 要发起与 B 的通信，可以直接通过 TCP 连接到 B。但如果 B 尝试通过 TCP 连接到 A 进行 P2P 通信，则会失败，原因是 A 位于 NAT 设备后，虽然 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， B 不是直接向 A 发起连接，而是通过服务器给 A 转发一个连接请求，反过来请求 A 连接到 B（即进行反向链接），A 在收到从服务器转发过来的请求以后，会主动向 B 发起一个 TCP 的连接请求，这样在 NAT 设备上就会建立起关于这个连接的相关表项，使 A 和 B 之间能够正常通信，从而建立起它们之间的 TCP 连接。

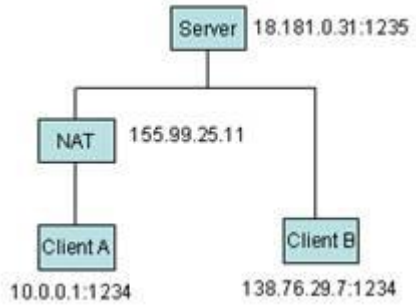


图 1 反向链接示意图

二、 UDP 打洞技术

适用场景： P2P 通信双方都位于 NAT 设备之后，且进行基于 UDP 应用的通信

UDP 打洞技术是通过中间服务器的协助在各自的 NAT 网关上建立相关的表项，使 P2P 连接的双方发送的报文能够直接穿透对方的 NAT 网关，从而实现 P2P 客户端互连。如果两台位于 NAT 设备后面的 P2P 客户端希望在自己的 NAT 网关上打个洞，那么他们需要一个协助者——集中服务器，并且还需要一种用于打洞的 Session 建立机制。下面分三种情景介绍该机制的工作过程。

集中服务器

集中服务器本质上是一台被设置在公网上的服务器，建立 P2P 的双方都可以直接访问到这台服务器。位于 NAT 网关后面的客户端 A 和 B 都可以与一台已知的集中服务器建立连接，并通过这台集中服务器了解对方的信息并中转各自的信息。

同时集中服务器的另一个重要作用在于判断某个客户端是否在 NAT 网关之后。具体的方法是：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。

Session 建立机制

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：

（1）A 最初不知道如何向客户端 B 发起连接，于是 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。

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

（3）当 A 收到由集中服务器发来的包含 B 的外网和内网的地址二元组信息后，A 开始向 B 的地址二元组发送 UDP 数据包，并且 A 会自动锁定第一个给出响应的 B 的地址二元组。同理，当 B 收到由集中服务器发来的 A 的外网和内网地址二元组信息后，也会开始向 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 A 回应的地址二元组。由于 A 与 B 互相向对方发送 UDP 数据包的操作是异步的，所以 A 和 B 发送数据包的时间先后并没有时序要求。

1. P2P 的两个客户端位于同一个 NAT 设备后面，即位于同一内网中

这是最简单的一种情况（如图 4 所示）。客户端 A 和 B 分别与集中服务器建立 UDP 连接，经过 NAT 转换后，A 的公网端口被映射为 62000，B 的公网端口映射为 62005。

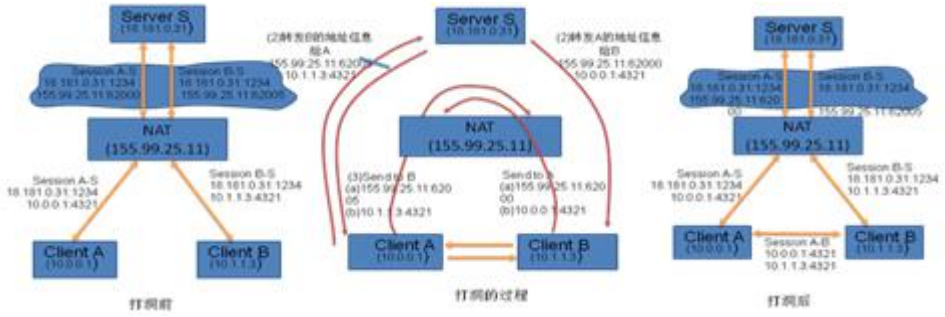


图 2 位于同一个 NAT 设备后的 UDP 打洞过程

当 A 向集中服务器发出消息请求与 B 进行连接，集中服务器将 B 的外网地址二元组以及内网地址二元组发给 A，同时把 A 的外网以及内网的地址二元组信息发给 B。A 和 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 A 与 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。A 与 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，P2P 双方也应忽略与内网地址二元组的连接，如果 A 和 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。

什么是 Hairpin 技术？

Hairpin 技术又被称为 Hairpin NAT、Loopback NAT 或 Hairpin Translation。Hairpin 技术需要 NAT 网关支持，它能够让两台位于同一台 NAT 网关后面的主机，通过对方的公网地址和端口相互访问，NAT 网关会根据一系列规则，将对内部主机发往其 NAT 公网 IP 地址的报文进行转换，并从私网接口发送给目标主机。目前有很多 NAT 设备不支持该技术，这种情况下，NAT 网关在一些特定场合下将会阻断 P2P 穿越 NAT 的行为，打洞的尝试是无法成功的。好在现在已经有越来越多的 NAT 设备商开始加入到对该转换的支持中来。

2. P2P 客户端位于不同的 NAT 设备后面，分属不同的内网

这是最普遍的一种情况（如图 3 所示）。客户端 A 与 B 经由各自的 NAT 设备与集中服务器建立 UDP 连接， A 与 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， A 的外网 IP 被映射为 155.99.25.11，外网端口为 62000；B 的外网 IP 被映射为 138.76.29.7，外网端口为 31000。

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

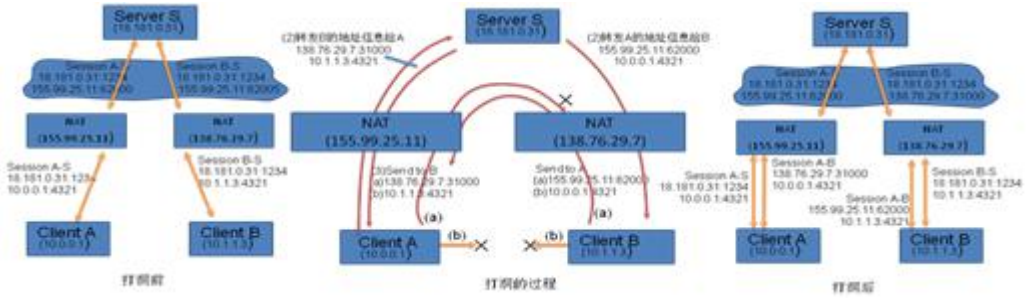


图 3 位于不同 NAT 设备后的 UDP 打洞过程

在 A 向服务器发送的登陆消息中，包含有 A 的内网地址二元组信息，即 10.0.0.1:4321；服务器会记录下 A 的内网地址二元组信息，同时会把自己观察到的 A 的外网地址二元组信息记录下来。同理，服务器也会记录下 B 的内网地址二元组信息和由服务器观察到的客户端 B 的外网地址二元组信息。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 和 B 分属不同的内网，它们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。当 A 的第一个消息发往 B 的外网地址（如图 3 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}，和 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，但它的目的地址是 B 的外网地址。在 A 的 NAT 设备支持保留 A 的内网地址二元组信息的情况下，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，这些数据包的外网地址二元组信息均被映射为{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，并丢弃该数据包。

B 发往 A 的消息包也会在 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。

3. P2P 客户端位于两层（或多层）NAT 设备之后，最上层的 NAT 设备通常是由网络提供商（ISP）提供，下层 NAT 设备是家用路由器

（如图 6 所示）假定 NAT C 是由 ISP 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内部网

络接入 NAT C 的内网，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（我们将这种由 ISP 提供的内网地址称之为“伪”公网地址）。同理，隶属于 NAT A 与 NAT B 的客户端，它们处于 NAT A，NAT B 的内网，以此类推，客户端可以放到到多层 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。

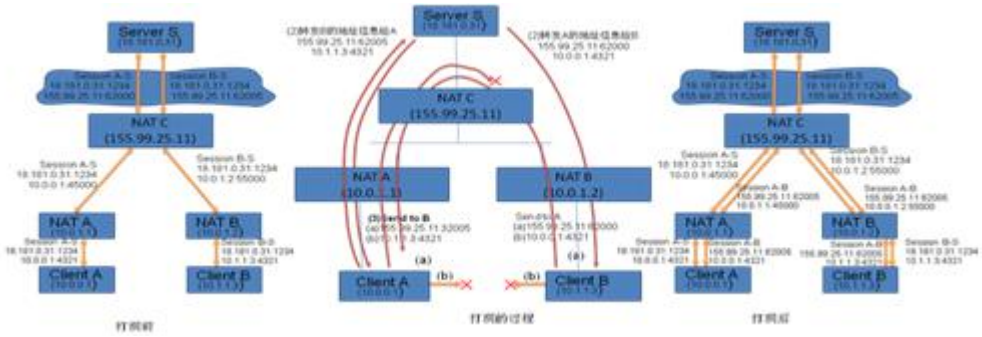


图 4 多层 NAT 下的打洞过程

假定客户端 A 和 B 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”公网地址（即{10.0.1.2:55000}）上发送数据包。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 真正的公网地址{155.99.25.11:62000}以及{155.99.25.11:62005}，但是客户端 A 与 B 无法通过服务器知道这些“伪”公网的地址；即使通过某种手段获知到了，我们仍然不建议采用上述的“最优化”的打洞方式，这是因为这些由 ISP 提供的地址有可能与客户端本身所在的内网地址重复，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，只能使用由集中服务器观察到的客户端 A、B 的公网地址二元组进行“打洞”操作，用于“打洞”的数据包由 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 数据包时，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 可以支持 Hairpin 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。

当然，从应用的角度上来说，在完成打洞过程的同时，还有一些技术问题需要解决，如 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”过程打出来的“洞”。如果 P2P 应用程序希望“洞”的存活时间不受 NAT 网关的限制，就最好在穿越 NAT 以后设定一个穿越的有效期。

对于有效期目前没有标准值，它与 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。

三、 TCP 打洞技术

从现在的主流应用的角度上来看，基于 TCP 的 P2P 应用显然不如基于 UDP 的应用那么广泛，但是也存在打洞的需求。

TCP 相对于 UDP 而言要复杂的多，TCP 连接的建立要依赖于三次握手的交互，所以 NAT 网关在处理 TCP 连接的时候，需要更多的开销。但是基于 TCP 的 P2P 打洞技术却比 UDP 复杂一点而已，而且，由于 TCP 协议完备的状态机机制，TCP 反而比 UDP 更能精确的获取某个 Session 的生命期。

在前面提到的各种场景下， TCP 打洞技术基本上也是沿用 UDP 打洞技术的思路。为了配合 TCP 在 NAT 上打洞，需要应用程序支持套接字和 TCP 端口重用技术。此后 P2P 通信双方只要各自向对方发起连接，以期待恰好一次尝试在双方的 NAT 网关上成功的打出一条通道。当然实际的打洞详细过程会考虑到很多特殊情况的处理方式，读者可参考相关文章参考，本文对此不赘述。

结束语

在 IP 地址极度短缺的今天，NAT 几乎已经是无所不在的一项技术了，以至于现在任何一项新技术都不得不考虑和 NAT 的兼容。作为当下应用最热门的技术之一，P2P 技术也必然面对 NAT 这个重镇。打洞技术看起来是一项近似乎蛮干的技术，却不失为一种有效的技术手段。在集中服务器的帮助下，P2P 的双方利用端口预测的技术在 NAT 网关上打出通道，从而实现 NAT 穿越，解决了 NAT 对于 P2P 的阻隔，为 P2P 技术在网络中更广泛的推广作出了非常大的贡献。