关于 P2P 内网穿越问题的研究

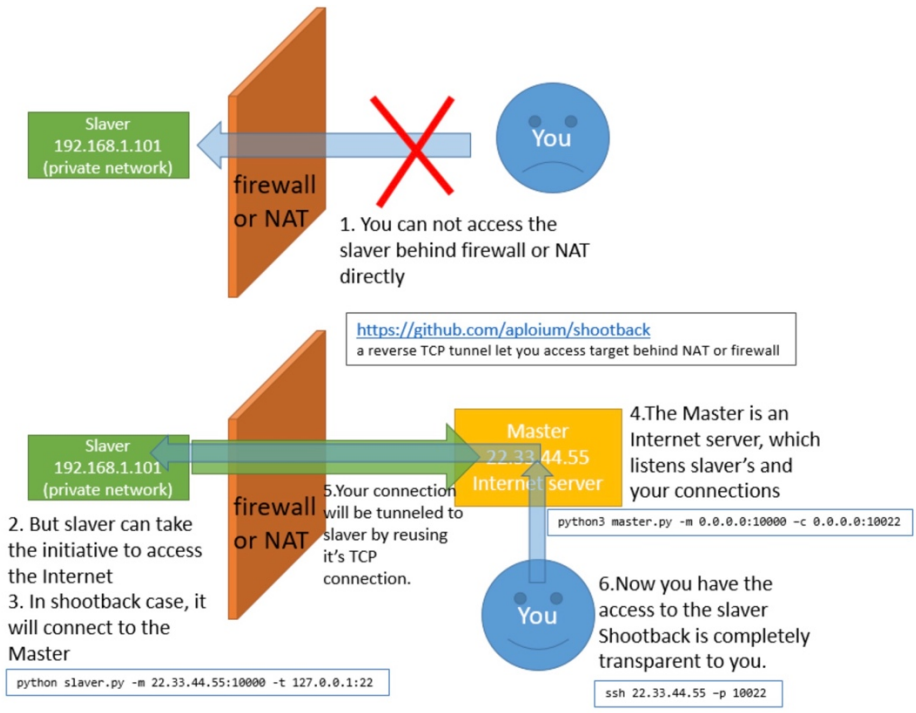
首先问题的关键在于：

1. 外网IP直接可以相互连接。
2. 处在同一内网下IP可以互联
3. 不同内网下的IP不能相互连接。
4. 内网和外网之间，只能从内网连接到外网，外网连接不到内网。

由于动态 NAPT 的映射关系是 LAN 侧数据包来触发的，如果 WAN 侧有主动进来的数据包，因为查询不到映射关系的存在，就会被丢弃掉。所以这时需要内网穿透。

1）方案一

理念是构建一个公网中的服务端，两个内网中的客户端都相当于向该服务器注册，然后发送三次握手，可互相发现，实现 NAT 穿透。Python 写的比较常用的实现 TCP 内网穿透的服务器是 ShootBack：



2）方案二

另一个比较简单的方式是：端口映射(Port Mapping 或者叫 Port Forwarding)，也是最基本的一种方式。

它将 NAPT 网关 WAN 侧的指定端口映射到内网指定地址的指定端口上。 这样当网关收到一个从外网过来的封包，就会转发到上述指定的内网地址和端口，对于封包的发起者来说，就像是直接访问这个内网主机一样。

且当内网的地址和端口都希望是动态的时候就需要 “动态端口映射”，也就是 UPnP 里面的 IGD（Internet Gateway Device）控制协议。

这种方式下的情形就是：把节点连接在某个 UPnP 分配的节点上，然后外部直接来连接这个端口。

现在了解到的 python 下分离并加强这种分配端口方法的的包就是 miniUpnp，还有待于了解；

但这种方法存在的问题：

——使用 UPNP 穿透 NAT 的方法会使得机器不安全，极其容易卡死路由器和其他问题；

——最初是为各种设备，后来发展为一般都是为电驴等软件配置某一个外部端口的时候开启，而自己写的某种程序能否做到端口映射问题很大，且开源多基于 C；

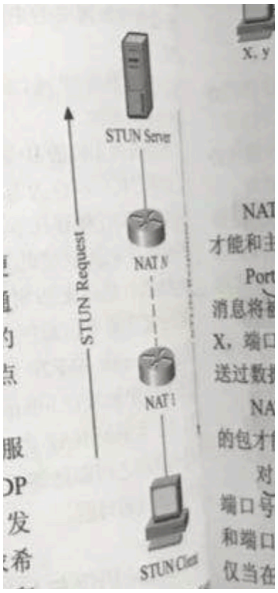
——这种方法受到不同网络状况和设备的影响很大；

——多用于家庭内的树莓派或者设备连接外部公司里电脑这种情况；

3）方案三

STUN 协议和 TURN 协议

STUN（Simple Traversal of UDP over NATs，NAT 的 UDP 简单穿越）是一种网络协议，它允许位于 NAT（或多重 NAT）后的客户端找出自己的公网地址，查出自己位于哪种类型的 NAT 之后以及 NAT 为某一 个本地端口所绑定的 Internet 端端口。这些信息被用来在两个同时处于 NAT 路由器之后的主机之间建立 UDP 通信。



如图，在 N 层 NAT 后面的 client 向 Server 询问自己的外网 IP 和端口号，然后直接用于连接，这样就可以直接使用外部端口和 IP，避免使用私网 IP 和端口无法被访问的问题。

网上开源项目也很多：

PyStun

A Python STUN client for getting NAT type and external IP

This is a fork of pystun originally created by gaohawk (http://code.google.com/p/pystun/)

PyStun follows RFC 3489: http://www.ietf.org/rfc/rfc3489.txt

A server following STUN-bis hasn't been found on internet so RFC3489 is the only implementation.