关于 P2P 内网穿越问题的研究（二）

1）一个可能存在的问题

首先，一个前提：如果各节点使用 IP（内网或公网）进行访问。

那么我自身测试 TCP 和 UDP 连接的过程中发现当前系统部署在不同节点（使用之前的四台 Ubuntu 服务器上）的情况下，如果用 IP（内网或者公网）访问（相对于现在虚拟机使用 127.0.0.1 这种方式来说）：

需要将绑定端口时的本地回环地址 127.0.0.1 改为 0.0.0.0，否则会发生 connection refused 的情况，TCP 与 UDP 都是这样。

2）Pynat 方法的测试

在各节点使用内网 IP 进行 TCP 和 UDP 沟通没有问题的情况下，我使用 Pynat 得到的公网 IP 测试。

这里要说明一下，Pynat 是可以设置源 IP 地址（内网）和端口（不是默认的 54320），这样就避免了端口的重复。

结果是 TCP 没有得到回应，我之后研究了 Pynat 的源码，排除了校园网防火墙的可能性（其三次 UDP 测试排除了这种可能性）。

还需要进一步研究其端口映射问题。

3）ShootBack 方法的测试

出于对比测试的考虑，我测试了在外网搭建中建索引服务器的方法，即之前提到过的 ShootBack 方法，是完全可行的，实现了穿越。

这种方法下，我在一个美国 IP 的 Ubuntu 服务器上搭建了索引服务器 Master，然后内部一个节点将自己的 TCP 监听端口设置为 0.0.0.0:9999，并将自身的端口设定告知服务器，外网或者内网的其他节点想要访问这个端口时，只需要向服务器的指定端口请求即可，测试成功。

```
[INFO 2019-01-24 08:14:59,381] Got slaver 59.64.129.75:35832 Total: 1
[INFO 2019-01-24 08:14:59,558] Got slaver 59.64.129.75:35834 Total: 2
[INFO 2019-01-24 08:14:59,732] Got slaver 59.64.129.75:35836 Total: 3
[INFO 2019-01-24 08:14:59,908] Got slaver 59.64.129.75:35838 Total: 4
[INFO 2019-01-24 08:15:00,083] Got slaver 59.64.129.75:35840 Total: 5
[INFO 2019-01-24 08:18:22,191] Serving customer: ('59.64.129.164', 53257) Total customers: 1
[INFO 2019-01-24 08:18:22,610] Got slaver 59.64.129.75:35926 Total: 5
[INFO 2019-01-24 08:19:42,421] customer complete: ('59.64.129.164', 53257)
[INFO 2019-01-24 08:22:10,749] Serving customer: ('59.64.129.164', 53273) Total customers: 1
[INFO 2019-01-24 08:22:11,150] Got slaver 59.64.129.75:36022 Total: 5
[INFO 2019-01-24 08:22:16,750] customer complete: ('59.64.129.164', 53273)
[INFO 2019-01-24 08:36:10,432] Serving customer: ('104.248.211.30', 38542) Total customers: 1
[INFO 2019-01-24 08:36:10,840] Got slaver 59.64.129.75:36368 Total: 5
[INFO 2019-01-24 08:36:15,641] customer complete: ('104.248.211.30', 38542)
```

上图中，是对内网的端口做了 5 个预设映射，前几次请求都是来自内网我自己电脑上的 TCP Client 的请求，得到了处理，之后我又尝试了另一台外网服务器（不同于这台索引服务器）的请求，也得到了处理。

结论 :ShootBack 实现穿越完全可行，但是缺点是映射路由需要手动写，所以主要用于将内网的网站站点暴露出去，被外界请求，是这种内网穿透情景下的最优解，对于我们这个系统来说未必最佳。

4）问题与下一步研究

实际上流行的内网穿越主要的文献和资料都是关于这种基于 ShootBack 的穿越方式，是绝对可以使外网访问内网指定节点的，但

是相对于 Pynat 过于繁琐。

但是问题在于，Pynat 的方式过于简单，STUN（即 Pynat 依靠的协议）已经提出很久了，在内网情境下是否可以使用还有很大问题，下一步任务是继续研究其源码，看看能否提取出适合系统情景的函数与包。而且最基本的要先测试成功，对此我个人存疑，希望能通过对比 SootBack 解决问题。