

使用OpenVPN连通多个局域网的一种配置方案



潘杰

9月7日 天津

📖 阅读 7 分钟

使用OpenVPN有一段时间了，它友好快捷的搭建了异地局域网的问题，在使用的过程中对配置的理解还是很重要的，在此以tun模式为例，说明其在实际使用过程的作用与意义。

常规配置

首先我们讲下都会讲到的常规配置，由于基本上所有的文章都会讲到，所以在此对部分常用的进行简单阐述：

port

说明服务运行的端口号，该端口号不能被其它的应用占用，同时需要在防火墙（如有）中做好允许访问的策略。

示例：

```
port 1194
```

udp

协议，可选为tcp和udp，由于udp的速度更快，一般默认使用该选项：

```
proto udp
```

dev

dev猜想应该是device设备的简称，这里指把OpenVPN做为一个什么类型的网络设备，分别可以设置成服务于3层的TUN（Tunnel）设备，以及服务于2层的TAP（Terminal Access Point）设备。

本文主要阐述 TUN 模式，后面的主要配置其实也是围绕着3层的路由配置来进行的。

ca、cert、key

这里涉及到一些非对称加密的知识，简单来讲就是密钥会成对出现。一个自己保留的叫做私钥，另一个公开传给其它计算机的叫做公钥。

使用公钥进行加密后的数据只能通过私钥进行解密，同样通过私钥进行加密的数据则需要只能公钥来进行解密。

由于公钥与私钥的互相解密的性质，所以它们必然成对出现。

在客户端（假设名称为client）为1个的时候，会生成3个密钥对。

1. 服务端私钥server.key，以及带有签名信息的服务端公钥server.crt
2. 客户端私钥client.key，以及带有签名信息的客户端公钥client.crt
3. 用于对公钥生成签名信息的私钥ca.key，以及用于验证签名信息的公钥ca.crt

OpenVPN的连接验证大概是这样：

1. 客户端向服务端传送自己的公钥client.crt，服务端接收到使用ca.crt验证client.crt签名是否正确，不正确禁止连接。
2. 客户端获取服务端公钥server.crt，使用使用ca.crt验证server.crt签名是否正确，不正确终止连接。
3. 通过各自的公钥私钥发送一些测试信息，以保证各自的秘钥对是正确的。

所以对于服务端而言，在安全认证方法，需要设置以下3个信息：

1. 用于验证证书签名是否正确的 ca.crt
2. 用于发送加密验证信息的私钥server.key
3. 用于客户端获取后解密测试数据的公钥server.crt

这套理论也适用于客户端。

示例配置：

```
ca /usr/local/etc/openvpn/keys/ca.crt
cert /usr/local/etc/openvpn/keys/openvpn-server.crt
key /usr/local/etc/openvpn/keys/openvpn-server.key
```

dh

dh指的是Diffie-Hellman算法，这个算法有个神奇的地方在于可以在大庭广众下商量一个只有两个人知道的密钥。

过程简单如下：

1. 服务端初始化两个数字： $p = 23$ 和 $g = 5$ （实际会大的多，本例中dh.pem中实际存储的就是23与5两个数字）
2. 客户端请求服务端获取到这两个数字： $p = 23$ 和 $g = 5$
3. 客户端随机选择一个整数，比如6，然后计算 $(g^a) \bmod p$ 并得到17。
4. 服务端也随机选择一个整数，比如 $b = 15$ ，计算 $(g^b) \bmod p$ 得到19。
5. 客户端把17传给服务端，服务端把19传给客户端。
6. 客户端 $19^6 \bmod 23 = 2$
7. 服务端 $17^{15} \bmod 23 = 2$

此时，服务端与客户端同时得到2这个数字，2即为服务端与客户端后面在通讯过程中使用的密钥。上述算法由于客户端使用的6与服务端使用的15并没有进行过任何传递，所以除客户端与服务端外，第三方无法计算出密钥2。

示例配置：

```
dh /usr/local/etc/openvpn/keys/dh.pem
```

ta.key

ta.key主要是提供了额外的认证机构，在进行认证时，除了进行签名认证，使用密钥对进行测试信息的加密解密外，还要求客户端与服务端的ta.key的值是一致的。需要说明的是：ta.key是一种对称密钥，也就是说：双方使用ta.key加密的数据均可以使用ta.key解密。 ta.key在服务器上存一份的同时，在每个客户端上均需要存储一份。

```
tls-auth /usr/local/etc/openvpn/keys/ta.key 0 # This file is secret
```

server

server设置了OpenVPN的服务网段，以下例配置为例：

```
server 10.8.0.0 255.255.255.0
```

则服务端的IP 将被设置为 10.8.0.1，然后客户端 IP也将设置为10.8.0.0/24段中的唯一 IP。这里需要根据客户端的数量来设置子网掩码的长度。在tun模式下，一个TUN会占用两个 IP，也就是说启用一个客户端则会占用两个 IP，所以在24位掩码（255.255.255.0）的情况下，大概可以连接125个客户端。

ifconfig-pool-persist

ifconfig = 网卡配置

pool = 池

persist = 持久化

所以它的意思是说固定客户端的 IP 地址，比如客户端client 在第一次请求时获取的 IP 为 10.8.0.4，则在客户端client断开重连后，其 IP 地址仍然会被设置为 10.8.0.4

示例配置：

```
ifconfig-pool-persist ip.txt
```

核心配置

核心配置决定了客户端与服务端，客户端与客户端，服务端与客户端所在网络的计算机是否能够正常通读，所以把它们弄懂非常的有必要。

同时由于一般的PC做为客户端时，并不需要执行数据转发等操作，所以基本上0配置，所以在此我们主要阐述路由器做为客户端时的各项配置。

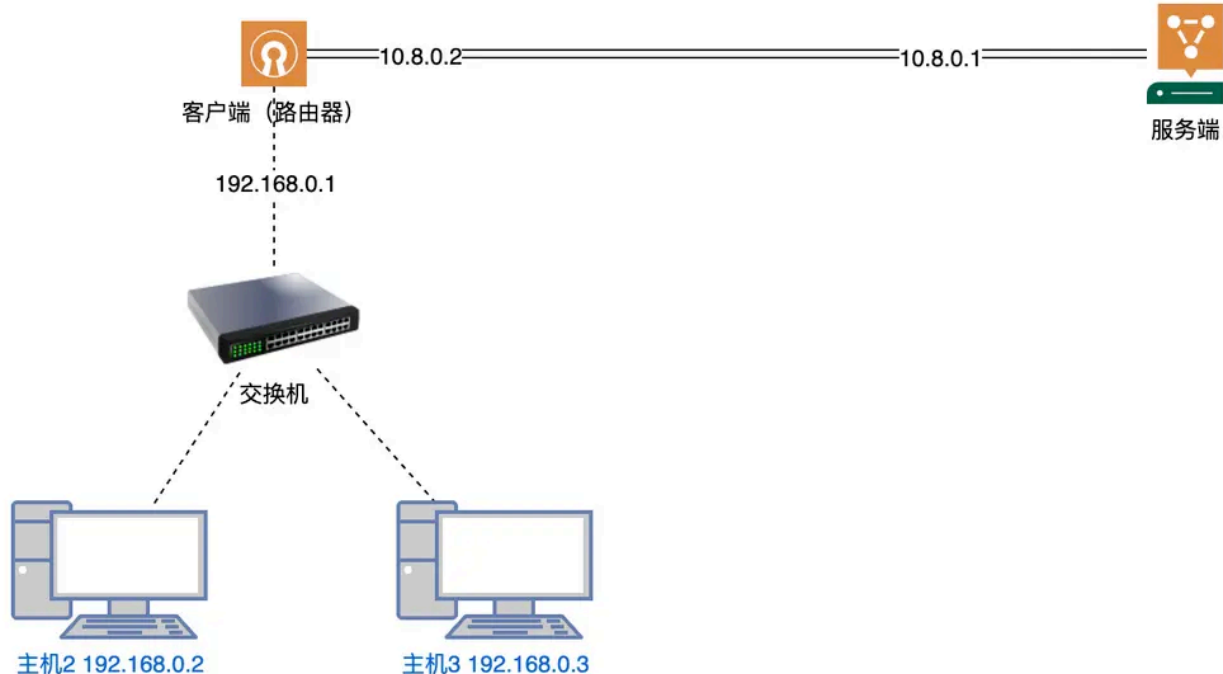
首先我们配置下ccd的目录，这样后面我们就可以为客户端进行一些简单的配置。

```
client-config-dir ccd
```

以下示例中，我们假设防火墙均已放行OpenVPN的连接。

仅连接路由器

拓扑图如下：



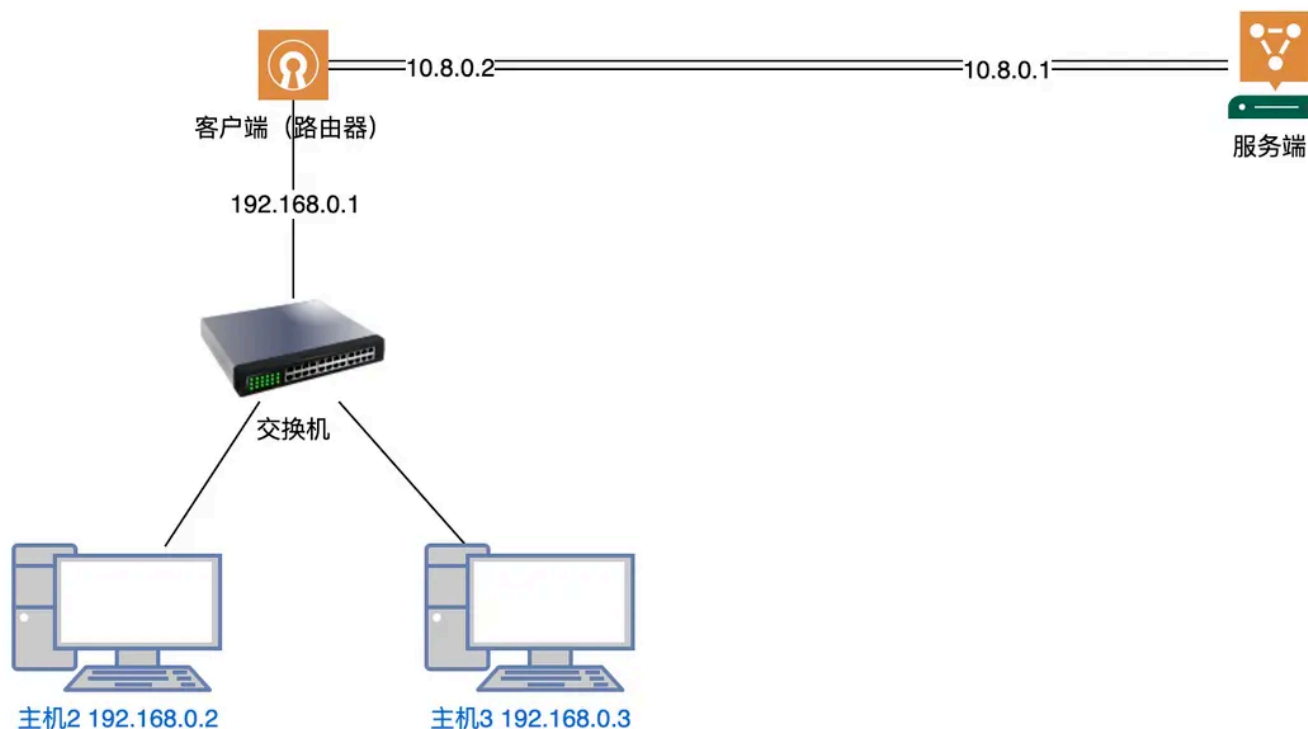
目标:

1. VPN服务端能够与路由器连通
2. VPN服务端不能够与主机2, 主机3连通

则不需要额外的配置, 客户端与服务端连接后, 设置好后则可以连接成功。

单客户端全连接

拓扑图如下:



目标:

1. VPN服务端能够与路由器连通
2. VPN服务端能够与主机2，主机3连通

配置路由

首先我们需要在服务端配置对应的路由：

```
route 192.168.0.0 255.255.255.0
```

此时，将在服务端宿主上添加路由：192.168.0.0/24 -> tun，即将192.168.0.0/24的请求转发给tun。

除此以外，并不需要其它配置。

当主机2向服务端10.8.0.1发送数据时，路径如下：

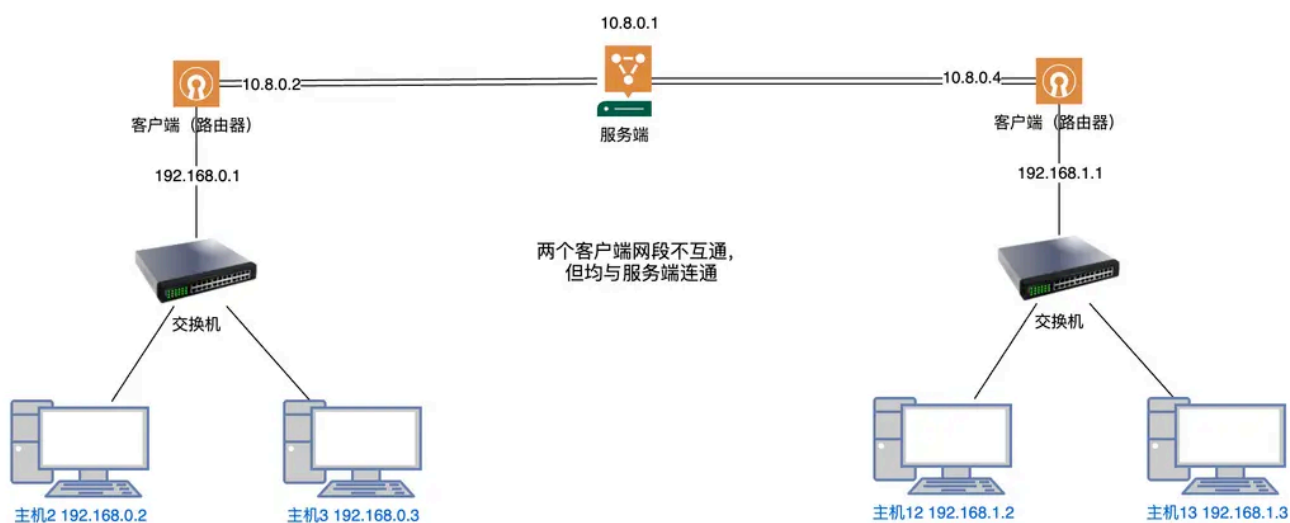
1. 主机2：发送给网关（路由器）192.168.0.1
2. 路由器：目标地址与tun路由匹配，发送给tun -> 10.8.0.1
3. 发送完毕

当服务端10.8.0.1向主机2发送数据时，路径如下：

1. 服务端：目标地址与路由192.168.0.0/24 -> tun匹配，发送给客户端（路由器）10.8.0.2
2. 路由器：目标地址为lan段，发送给lan段的主机2
3. 发送完毕

双独立客户端

拓扑图如下：



目标：

1. VPN服务端能够与两个路由器均连通
2. VPN服务端能够与主机2，主机3连通

3. VPN服务端能够与主机12，主机13连通
4. 主机2与主机3，路由器1与路由器2，均不连通。

配置路由

配置路由的方法与 全连接 的配置方法相同。

首先我们需要在服务端配置对应的路由：

```
route 192.168.0.0 255.255.255.0
```

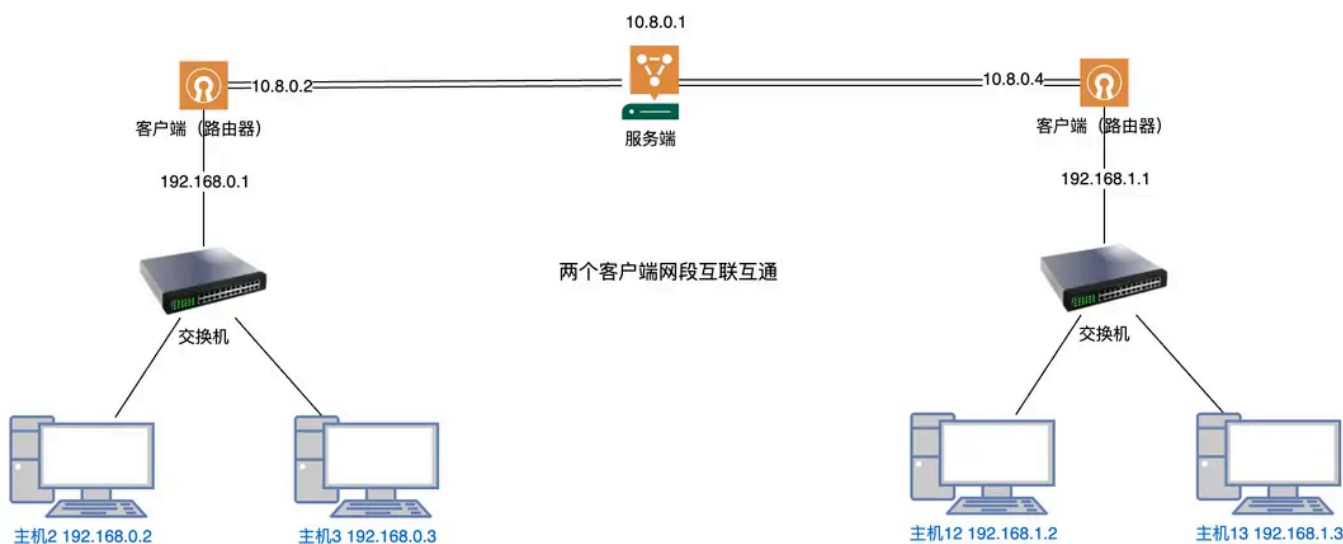
此时，将在服务端宿主上添加路由：192.168.0.0/24 -> tun，即将 192.168.0.0/24 的请求转发给 tun。

除此以外，并不需要其它配置。

此时两个局域网在进行数据发送时，仅当请求地址为192.168.0.1，才会走tun，所以不会对其它的产生影响。

双客户端互连互通

更多的时候，我们可能需要的是双客户端互连互通



开启client-to-client

默认情况下client-to-client配置项并未启用，所以客户端间是不能够进行连接的，要开启该功能，需要先在配置文件中加入：

```
client-to-client
```

配置路由

然后告诉VPN服务器：除了 VPN服务端本身的网段需要通过 TUN 转发外，`192.168.0.0/24`以及`192.168.1.0/24`段也需要通过TUN进行转发，配置如下：

```
route 192.168.0.0 255.255.255.0
route 192.168.1.0 255.255.255.0
```

或者直接使用：

```
route 192.168.0.0 255.255.254.0
```

此时，服务端自身在进行`192.168.0.0/23`的请求时，便会把数据转发给tun

路由推送

两个网段互通的前提时：当向对向网段发起请求时，将数据包转发给tun。所以要在服务端配置下发给客户端的路由：

```
push "route 192.168.0.0 255.255.255.0"
push "route 192.168.1.0 255.255.255.0"
```

注意，这里不能写成`push "route 192.168.0.0 255.255.254.0"`，虽然在路由表中这种写法与上面两条的写法的作用是相同的。但在此不同。每个网段必须单独的写一条下发路由命令。

此时客户端在连接服务端后，则会在本地添加两条路由：

```
192.168.0.0/24 -> tun
192.168.1.0/24 -> tun
```

该路中的下发会引发客户端的路由地址冲突，因为客户端本身就存在`192.168.0.0/24`或`192.168.1.0/24`的路由段，该段的路由转发目的地为网关。会导致局域网访问不通的问题。

所以在配置推送（下发）的路由时，必须与ccd配置相结合。

ccd配置

ccd文件夹中存放客户端的配置，比如两个客户端名称分别为client, client1，对应0.0以1.0网段。

则在ccd需要创建文件：`client`，然后存放以下内容：

```
iroute 192.168.0.0 255.255.255.0
```

此配置有2个作用：

1. 为服务端添加路由表，在 VPN 服务端进行数据转发时，将192.168.0.0 255.255.255.0转发给 client
2. 在进行路由推送时，忽略192.168.0.0 255.255.255.0对client的推送。

所以此时client对应的路由器将仅仅得到下发的路由： 192.168.1.0 255.255.255.0

同时，还需要在ccd中创建client12：

```
iroute 192.168.1.0 255.255.255.0
```

最后：

路由器1得到的路由(示意，实际路由条件会多，但效果相同) 如下：

```
192.168.1.0/24 -> tun
10.8.0.1 -> tun
```

路由器1得到的路由(示意，实际路由条件会多，但效果相同) 如下：

```
192.168.0.0/24 -> tun
10.8.0.1 -> tun
```

服务端路由：

```
192.168.0.0/24 -> client
192.168.1.0/24 -> client1
```

数据流

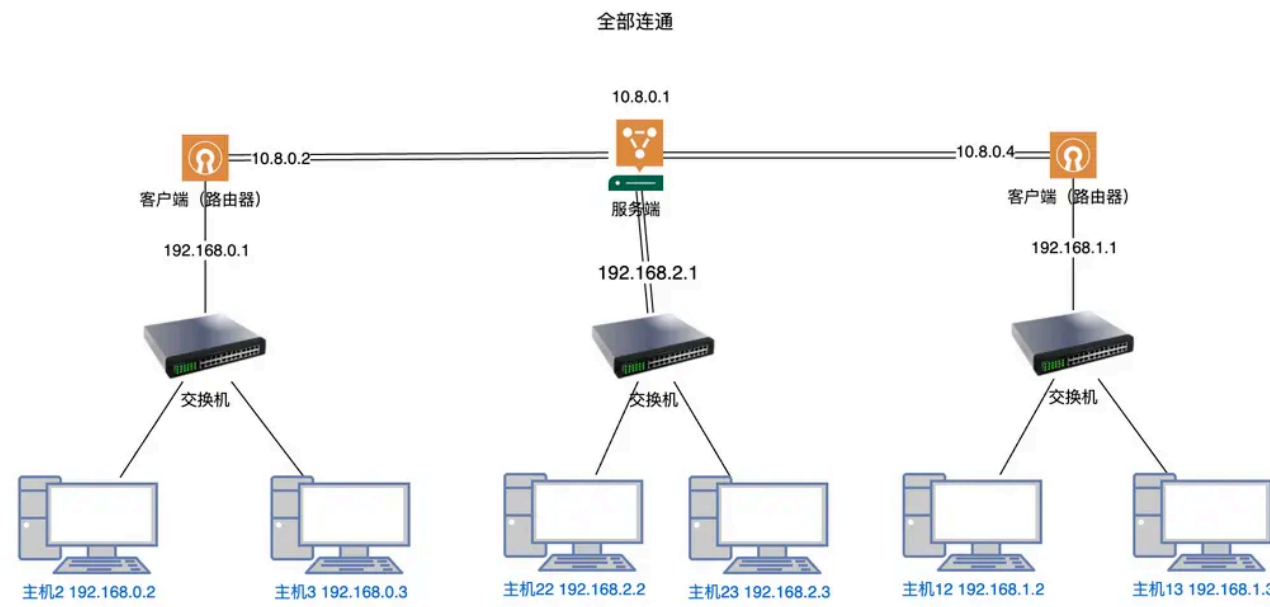
192.168.0.2 -> 192.168.1.2

1. 192.168.0.2将数据发送给网关192.168.0.1
2. 192.168.0.1接收到目标地址为 192.168.1.2 的数据包，根据路由表将其转发给VPN 服务端 10.8.0.1
3. 10.8.0.1接收到目标地址为 192.168.1.2 的数据包，根据路由表将其转发给client1路由器
4. client1路由器接收到目标地址为 192.168.1.2 的数据包，路由表显示其为局域网地址，直接转发给 192.168.1.2

192.168.1.2 -> 192.168.0.2 原理相同

全连通

还有时候，我们还需要连通服务端所在的局域网：



此时需要在 **双客户端互连互通** 配置的基础上，完善服务端的路由信息：

```
route 192.168.0.0 255.255.255.0
route 192.168.1.0 255.255.255.0
route 192.168.2.0 255.255.255.0
```

以及增加路由推送：

```
push "route 192.168.0.0 255.255.255.0"
push "route 192.168.1.0 255.255.255.0"
push "route 192.168.2.0 255.255.255.0"
```

则可以实现所有的客户端均可连接的目的。

示例配置

全连通的示例配置如下，该配置工作在OpenWRT下，如果工作在其它的操作系统上，注意替换user与group的值。

```
port 1194

proto udp

dev tun

ca /etc/openvpn/ca.crt
cert /etc/openvpn/openvpn-server.crt
key /etc/openvpn/openvpn-server.key # This file should be kept secret
dh /etc/openvpn/dh.pem
```

```
tls-auth /etc/openvpn/ta.key 0 # This file is secret

server 172.29.253.0 255.255.255.0

ifconfig-pool-persist ipp.txt

# push to client route
# lan
push "route 192.168.20.0 255.255.255.0"
# tute
push "route 192.168.8.0 255.255.248.0"
# tute -> vpn services
push "route 172.29.248.0 255.255.248.0"
# laoting
push "route 172.29.0.0 255.255.248.0"
```

ccd下文件配置如下：

客户端： laoting



```
iroute 172.29.0.0 255.255.248.0
```

客户端： tute

```
iroute 172.29.248.0 255.255.248.0
iroute 192.168.8.0 255.255.248.0
```

服务端路由表

Active IPv4 Routes

Device	Target	Gateway	Metric	Table	Protocol
wan	0.0.0.0/0		0	main	
wan		-	0	main	
(tun0)	172.29.0.0/21	172.29.253.2	0	main	
(tun0)	172.29.248.0/21	172.29.253.2	0	main	
(tun0)	172.29.253.0/24	172.29.253.2	0	main	
(tun0)	172.29.253.2	-	0	main	
(tun0)	192.168.8.0/21	172.29.253.2	0	main	
lan	192.168.20.0/24	-	0	main	

客户端laoting路由表

Active IPv4-Routes

Network	Target	IPv4 gateway	Metric	Table	Protocol
wan	0.0.0.0/0	172.29.1.1	0	main	
wan	172.29.1.0/24	-	0	main	
lan	172.29.2.0/24	-	0	main	
(tun0)	172.29.248.0/21	172.29.253.5	0	main	
(tun0)	172.29.253.0/24	172.29.253.5	0	main	
(tun0)	172.29.253.5	-	0	main	
(tun0)	192.168.8.0/21	172.29.253.5	0	main	
(tun0)	192.168.20.0/24	172.29.253.5	0	main	

客户端tute路由表

Network	Target	Gateway	Metric	Table	Protocol
wan	0.0.0.0/0	192.168.10.1	0	main	
(tun0)	172.29.0.0/21	172.29.253.9	0	main	
(tun0)	172.29.253.0/24	172.29.253.9	0	main	
(tun0)	172.29.253.9	-	0	main	
wan	192.168.10.0/24	-	0	main	
lan	192.168.11.0/24	192.168.12.254	0	main	
lan	192.168.12.0/24	-	0	main	
(tun0)	192.168.20.0/24	172.29.253.9	0	main	

客户端配置

忽略路由

在有些时候，我们并不希望向某些特殊的客户下发所有的路由的信息。这时候就可以在客户端的配置文件中加入：

```
pull-filter ignore "route"
```

来忽略到服务端下发的所有路由。

所以想忽略某些特定的路由，比如忽略: 192.168.1.0/24 则可以进行如下指定：

```
pull-filter ignore "route 192.168.1.0 255.255.255.0"
```

如果有个需要忽略的路由，则可以重复加上上述配置。

但这需要注意的是：被忽略的路由必须与服务端下发的路由规划完全一致，比如无法处理如下场景：

1. 服务端下发路由为：192.168.0.0 255.255.254.0
2. 客户端忽略路由为：192.168.0.0 255.255.255.0

虽然192.168.0.0 255.255.255.0是192.168.0.0 255.255.254.0的子路由，但OpenVPN客户端无法处理上述信息。

设置路由

除了可以忽略路由外，还可以在客户端的配置文件中手动的指定路由：

```
route 192.168.1.0 255.255.255.0
```

此时，在OpenVPN配置文件生效时，将会在路由表中添加192.168.1.0 255.255.255.0的出口为VPN。

总结

其实还可以根据配置信息完成更复杂的配置，比如将某个客户端所在网络的其它网络加入到整个VPN 大网络中。整体对配置总结如下：

1. 服务端路由的作用是配置当前 VPN 服务器的路由信息，在配置文件中声明的路由将加入到 VPN 服务端的路由表中，以达到当 VPN 服务器接收到目标地址的数据转发时，将数据交给 VPN 服务来处理。
2. 路由下发中配置的路由将下发给客户端，客户端在接收到下发的路由后，更新本地路由表，符合路由表的数据将转发给VPN 服务端
3. ccd下的文件必须与客户端名称相同。在其中可以定义客户端所在的网段信息（可以是多个），它的作用有两个：一是告之 VPN 服务端，当目标地址符合配置的路由时，将数据转发给特定的客户端。二是在向客户端进行路由下发时，忽略配置中的路由地址以避免发生路由冲突。

另外，即使不启用client-to-client，路由的下发也是生效的，此时就会有尴尬的事情发生：数据按路由指示转发给了VPN 服务端，但由于 VPN 服务端并没有开启client-to-client，从而使得通讯失败。所以如果在服务端上想关闭client-to-client功能，则需要同时删除到配置的所有路由推送。