## 一、VPN与SS区别

### Shadowsocks

Shadowsocks(简称ss)原理是socks5代理，socks代理只是简单的传递数据包，而不必关心是何种协议，所以socks代理比其他应用层代理要快的多。

Socks5代理是把你的网络数据请求通过代理，由服务器转发到目的地，这个过程中你是没有通过一条专用通道的，只是数据包的发出，然后被代理服务器收到，整个过程并没有额外的处理。

通俗的说，现在你有一个代理服务器在美国，比如你现在想要访问google，你的电脑发出请求，流量通过socks5连接发到你在美国的服务器上，然后再由你在美国的服务器去访问google，再把访问结果传回你的电脑，这样就实现了FQ。

### Vpn

vpn--虚拟专用网络，功能是：在公用网络上建立专用网络，进行加密通讯。在企业网络和高校的网络中应用很广泛。你接入vpn，其实就是接入了一个专有网络，你的网络访问都从这个出口出去。

vpn最主要的功能并不是用来FQ，只是它可以达到FQ的目的。

## 二、WireGuard介绍

WireGuard是Jason A. Donenfeld开发的开源VPN协议。目前支持Linux, macOS, Android以及OpenWrt。

## 三、WireGuard优点

1、跨平台

2、使用 UDP 传输数据，轻量、高效、节能、节省带宽

3、效率极高，WireGuard 协议作为 Linux 内核模块运行

4、免费开源

5、官方客户端可以允许部分应用绕过VPN

## 四、WireGuard vpn原理

服务器和客户端都安装一样的wireguard软件，两者地位是平等的。然后在配置文件中添加彼此的private key和public key，以及udp监听的端口、服务器地址。从原理上讲，谁先主动发起传输请求，谁就是服务器端，没有多少区别。

实际应用中VPS服务器有固定IP地址，我们把VPS当成Server端，手机为client端，client发送请求连接到server端，网络数据请求从server端出去，实现FQ目的。

### 配置解析

Server端主要配置：

**[Interface]**

**PrivateKey = server端私钥**

**Address = server端内网ip**

**ListenPort = server端UDP端口**

**[Peer]**

**PublicKey = client端公钥**

**AllowedIPs = 该账号client端内网ip**

**[Peer]**

**…另一个账号**

**[Peer]**

**…另一个账号**

Clent端主要配置：

[Interface]

PrivateKey = client端私钥

Address = client端内网ip

[Peer]

PublicKey = server端公钥

Endpoint = 服务器公网地址和端口

AllowedIPs= 允许连接ip，客户端一般写全部ip段 0.0.0.0/0, ::0/0

### 官方示例：

Client A

内网IP：10.0.0.2

Server B

公网ip：1.2.3.4:22 真实可用的ip端口

内网ip：10.0.0.1

当一个client A发送数据给server B：

1. 如果一个数据包要发送给10.0.0.1（**B的虚拟内网地址**）。WireGuard发送端先会看10.0.0.1是否已注册。如果未注册，数据包直接丢掉。
2. 如果B已经注册，使用B的公钥加密数据包。
3. 再看数据包要发送的**外网真实地址和端口**，如果和B的不匹配，则数据包直接丢掉。
4. 以上检查完毕，发送数据包。

当一个server B收到client A的数据：

1. 监听的端口收到数据包，准备解密。
2. 用B的私钥解密，检查数据中包含的B的外网地址和端口，以确定是A发送给B的。
3. 检查数据包中A的虚拟内网地址。
4. 如果A的虚拟内网地址未在B处注册，则直接丢掉，接受该数据包。

## 五、服务器二次开发

### 1、资源分发及回收

WireGuard服务端，需要支持多客户端连接(支持1000左右用户)

IP分配及回收策略：

**每个客户端连接的时候都需要服务端给分配一个ip，同时发送自己的公钥给服务器，客户端关闭时(断开连接)回收这个ip。**

设计思路：

#### IP资源池

设计使用两个redis集合进行ip资源维护

**ip\_available\_set** : 可用IP

**ip\_used\_set** : 已用IP + 客户端公钥

1. 上面两个redis集合，ip\_available\_set只存放ip地址，ip\_used\_set 存放已用ip + 客户端的公钥信息。
2. 服务端启动的时候初始化所有的可用ip地址，存入ip\_available\_set。
3. 客户端的连接请求中有**2**个参数，**上一次连接ip**和**客户端公钥**，如果是第一次连接，ip参数为空。
4. 服务器给客户端的连接响应也包括**2**个参数，**客户端ip**和**服务器公钥**，服务器从ip\_available\_set (spop)取出一个ip，将此ip和客户端公钥进行组合，写入 ip\_used\_set (sadd) 标记为已用，并把这个ip和服务器公钥响应给客户端。
5. 客户端非第一次请求建立连接，请求中携带上一次连接使用过的ip和客户端公钥，服务器检测此ip和客户端公钥的当前状态，检测此ip是否在ip\_available\_set集合中：

1)、如果存在，说明此ip已释放，当前可用状态，服务器执行第4步的步骤给客户端重新分配ip

2)、如果不存在，说明此ip当前已用，继续将此ip和客户端公钥进行组合，并在ip\_used\_set集合中查询是否存在

A、如果存在说明此ip使用者就是这个客户端，直接将此ip和服务器公钥响应给客户端。

B、如果不存在，说明此ip当前其他客户端在使用，服务器执行第4步步骤给客户端重新分配ip。

1. 客户端请求断开连接，请求中附带**ip**和**客户端公钥**2个参数，服务器将此ip和客户端公钥进行组合，从 ip\_used\_set (srem) 删除这个组合

1)、如果删除成功，同时在 ip\_available\_set (sadd) 添加此ip，使此ip变为可用。

2）、如果删除失败，说明ip\_available\_set 中不存在此记录，忽略本次操作。

1. 由于客户端不能确保每次退出时都发送断开连接请求，导致ip资源一直被占用，需要定时回收资源，服务端定期(定时任务) 采用ping的方式对ip\_used\_set 集合的所有元素的ip进行检测，不能ping通的 从ip\_used\_set(srem) 集合删除，并把此ip添加到ip\_available\_set (sadd)集合，释放ip资源。
2. 若ip\_available\_set 集合为空，客户端请求建立连接，需要分配ip的时候，提示此服务器资源已用尽(code=400)，请连接其他服务器。
3. 客户端在没有退出的状态下，由于某种原因资源被回收。此时客户端检测到不可用时，发送请求到服务器，请求新的资源。

#### WireGuard配置文件

采用动态方式添加删除客户端

##### 建立连接

1. 客户端建立连接时，首先自己生成publickey(公钥) privatekey(私钥)，请求中携带publickey发送到服务器
2. 服务器收到请求，根据ip资源池规则给客户端分配ip，并在服务器上执行添加客户端配置代码:

wg set wg0 peer cpublickey allowed-ips cip/32

wg-quick save wg0

最后将ip和服务器公钥下发给客户端。

##### 断开连接

1. 客户端断开连接时，请求中携带ip和publickey参数，发送到服务器
2. 服务器收到请求，在服务器上执行删除客户端配置代码:

wg set wg0 peer cpublickey remove

wg-quick save wg0

1. 根据ip资源池规则释放此ip

### 2、serverlist开发

Client 使用Wireguard方式，服务器分发不同的服务器列表

脚本

Git.io/fptwc