

翻墙代理 - Fireplusplus的博客 - 博客频道

•



版权声明：本文为博主原创文章，转载需注明出处。

目录(?)[\[+\]](#)

引言

(声明：纯技术博客！)

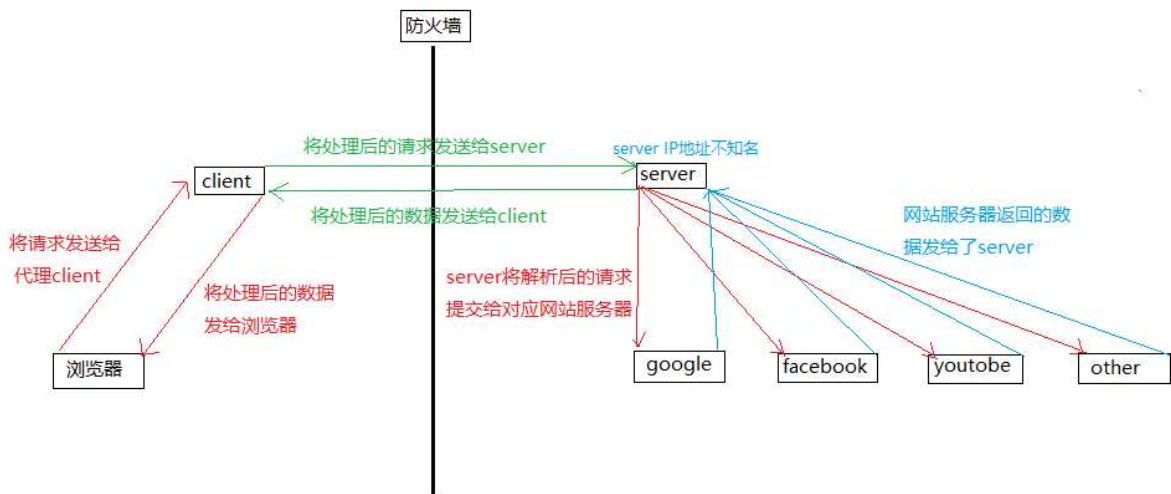
曾几何时，在国内还是可以访问google、faceboook、youtube之类的网站的，后来由于防火墙的原因，导致我们不再能够访问这些网站了，不免为人生一大憾事！后来出于各种各样的需求，例如查找学习资料等，就有人想要翻墙了，于是就有了翻墙代理。

原理

翻墙代理实现的关键在于如何绕过防火墙，得先知道防火墙是如何阻止我们访问外网的？大致有以下三种手段：

- 1. 域名劫持--->当你的主机发出DNS请求地址解析服务，就给你返回一个错误的ip或者压根不返回
- 2. IP黑名单--->国家入口网关封锁该IP，数据无法通过
- 3. 主干路由器关键字过滤阻断--->含有敏感关键字一率不允许通过

先看原理图再解释：



过程：

客户端负责充当一个跳板，将浏览器的请求数据加密（目的是让数据不能被直接识别）后转发给server。由于数据经过了加密，所以也就不会被主干路由器的关键字屏蔽技术阻挡。我们直接让client与server沟

通，所以用不到DNS服务，也就不会存在被假的IP欺骗或是找不到IP地址的情况。又因为我们会给server选择一个不知名的IP地址，即不会是防火墙IP黑名单中的地址，所以不会被封锁。（跨出国门第一步）

server接收到来自client的数据，先解密，然后代理浏览器发出请给对应网站服务器。网站服务器将server请求的数据再传给它，server再将数据进行加密，发送给我们的client。

最后，client将解密后的数据交给浏览器，这便是大致的原理了。

SOCKS5协议：

SOCKS5 是一个代理协议，它在使用TCP/IP协议通讯的前端机器和服务器机器之间扮演一个中介角色，使得内部网中的前端机器变得能够访问Internet网中的服务器，或者使通讯更加安全。SOCKS5 服务器通过将前端发来的请求转发给真正的目标服务器，模拟了一个前端的行为。在这里，前端和SOCKS5之间也是通过TCP/IP协议进行通讯，前端将原本要发送给真正服务器的请求发送给SOCKS5服务器，然后SOCKS5服务器将请求转发给真正的服务器。

我们这里也要借助这个协议，让我们的数据通过防火墙。下面列出相关的知识。

RFC1928（URL为：<http://www.ietf.org/rfc/rfc1928.txt>）描述了Socks协议的细节，告诉我们客户程序如何同Socks代理协商，取得透过该协议对外传输的途径。看了看RFC1928文档，大概整理出了socks5服务器处理TCP请求的流程：

socks5连接建立的时候首先由客户端发出如下格式数据给服务端：

VER	NMETHODS	METHODS
1	1	1 to 255

在SOCKS5中VER字节总是0x05代表socks5，NMETHODS存放客户端可以接受的认证方法数量n，后面的n个字节就是各个方法对应的编号了。

收到客户端发出的请求后，服务端做出回应：

VER	METHOD
1	1

其中验证方法对应的编号如下：

- o X' 00' NO AUTHENTICATION REQUIRED
- o X' 01' GSSAPI
- o X' 02' USERNAME/PASSWORD
- o X' 03' to X' 7F' IANA ASSIGNED
- o X' 80' to X' FE' RESERVED FOR PRIVATE METHODS
- o X' FF' NO ACCEPTABLE METHODS

客户端接受到如此回应后就可以开始与服务端进行验证了。在此0x02号代表的用户名密码验证方式在RFC1929文档中有介绍。我先不管验证，先实现匿名socks5代理。以后再考虑添加验证功能。

验证完毕后，客户端就可以开始提出要求了。到底要socks5代理为它做什么。请求数据包的格式如下：

VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X'00'	1	Variable	2

各字节代表的意义：

- o VER protocol version: X'05'
- o CMD
 - o CONNECT X'01'
 - o BIND X'02'
 - o UDP ASSOCIATE X'03'
- o RSV RESERVED
- o ATYP address type of following address
 - o IP V4 address: X'01'
 - o DOMAINNAME: X'03'
 - o IP V6 address: X'04'
- o DST.ADDR desired destination address
- o DST.PORT desired destination port in network octet order

这里的目标地址可以有三种格式(IPV4 IP, 域名, IPV6 IP)。其中IPV4, IPV6地址固定长度分别为4字节, 16字节。域名以pascal字符串保存：第一字节为字符串长度，后面跟字符串内容。

收到客户端的请求后, 服务端开始回应：

VER	REP	RSV	ATYP	BND.ADDR	BND.PORT
1	1	X'00'	1	Variable	2

Where:

- o VER protocol version: X'05'
- o REP Reply field:
 - o X'00' succeeded
 - o X'01' general SOCKS server failure
 - o X'02' connection not allowed by ruleset
 - o X'03' Network unreachable
 - o X'04' Host unreachable
 - o X'05' Connection refused
 - o X'06' TTL expired
 - o X'07' Command not supported
 - o X'08' Address type not supported
 - o X'09' to X'FF' unassigned
- o RSV RESERVED

- o ATYP address type of following address
 - o IP V4 address: X' 01'
 - o DOMAINNAME: X' 03'
 - o IP V6 address: X' 04'
- o BND. ADDR server bound address
- o BND. PORT server bound port in network octet order

Fields marked RESERVED (RSV) must be set to X' 00'.

对应客户端的3种不同要求:

针对CONNECT请求的回复:

BND. PORT里面保存的是服务端连接到远程服务器所分配的端口, BND. ADDR保存的是分配的IP地址。(这里socks5为什么这么做我还有点不清楚, 不过不管它, 先实现再说, 以后在看看socks5 proxy client那边是怎么利用的)

针对BIND请求的回复:

服务端将两次回复客户端。第一次回复是在服务端开启新的socket并开启一个端口侦听后。BND. PORT存放服务端侦听的端口。BND. ADDR存放IP地址。客户端一般会利用这段信息来告诉远端服务器。第二次回复是用来表明远端服务器连接成功。BND. PORT和BND. ADDR分别存放远端服务器连接用的端口及IP。

源码

源码挺长的, 就不贴出来了, 托管在GitHub上, 附上链

接: <https://github.com/Fireplusplus/Project/tree/master/esocks>

启动步骤

(费尽心思从别人那里借来的服务器的账号, 当然你也可以自己买一个: 网址: <https://www.linode.com/>就是有点小贵, 一个月大概60~70RMB)

首先, 得把server部署到服务器上:

```
scp server user_name@IP:/dir ---> 将服务器应用程序拷到远端服务器上
```

```
ssh user_name@IP ---> 登录到服务器
```

```
./ser 6666 ---> 将服务器跑起来 (6666就是client与server通信的端口, 不冲突即可)
```

启动客户端:

```
./cli 6667 IP 6666 ---> 第一个端口号为client与浏览器通信端口, 后边接服务器IP和端口号
```

最后，设置浏览器信息（以FireFox浏览器为例）：

选项--->高级--->网络--->设置：



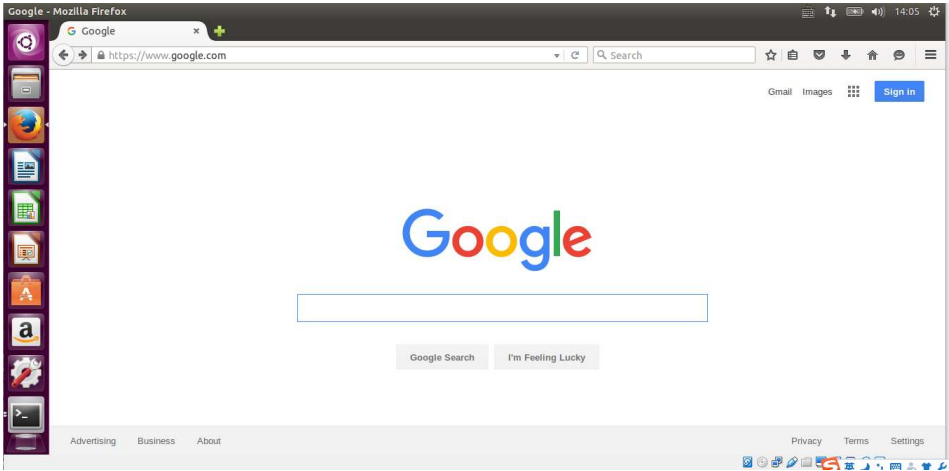
运行结果

现在就可以访问外网啦！

熟悉的界面又回来了



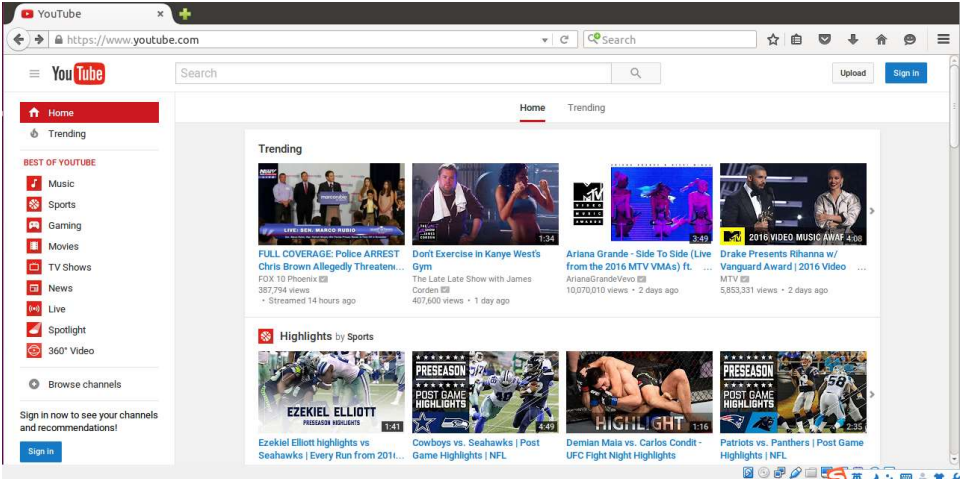
:

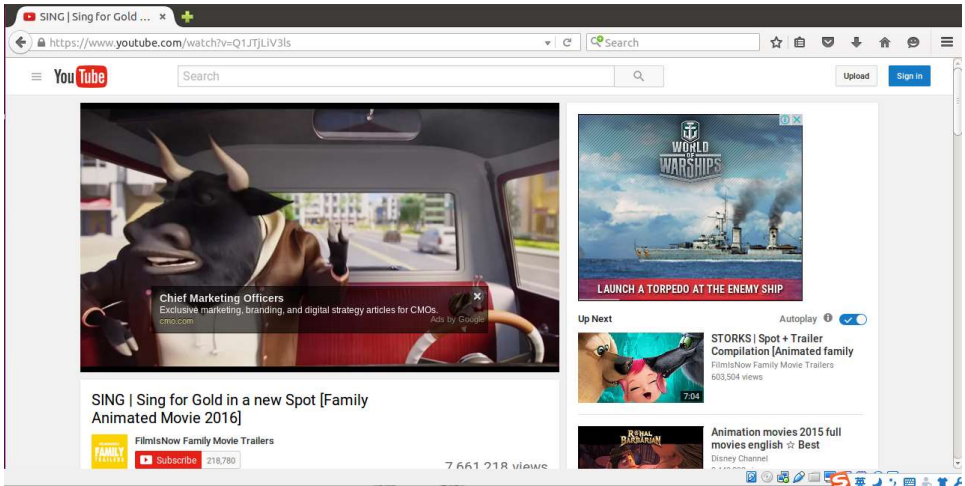


上youtube看个小视频



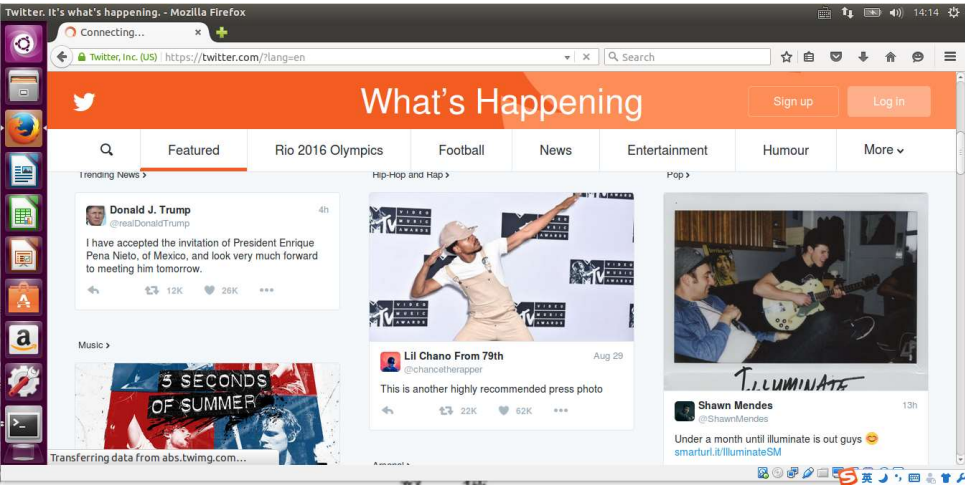
:





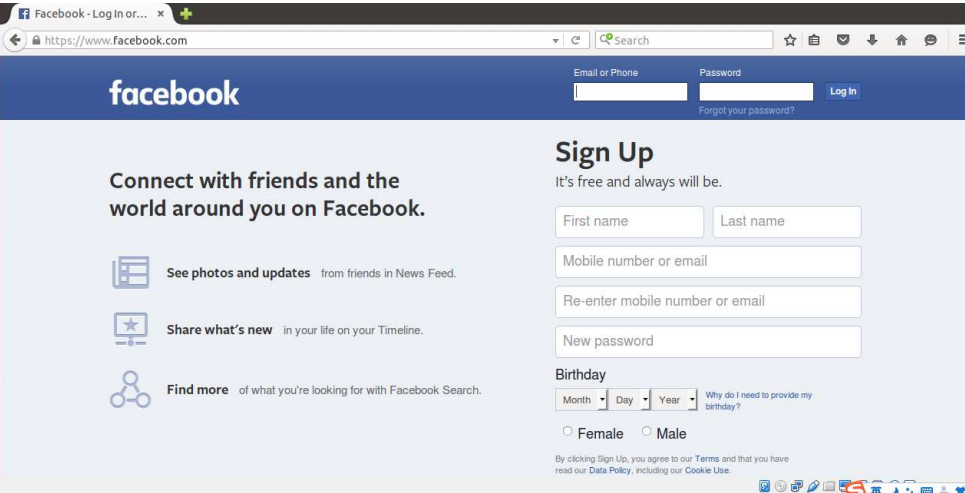
画质清晰，视频流畅，nice!

上个Twitter:



国外的微博，不错呦！

再来个Facebook:



嫌麻烦，就不注册了。

遇到的问题

1. server与client无法建立连接，提示验证方法错误。仔细分析后发现是加密解密对应关系没搞好，导致server未能识别client请求。重新修改了加密和解密的位置。

2. 浏览器设置问题，client与server运行起来，但依旧无法访问外网。解决办法就是一个字：试。幸好选项不多，多次试验下来，搞定。设置方法在前面已经贴出。

- 原创：96篇
- 转载：16篇
- 译文：0篇
- 评论：9条