科学上网目前大致分为这几类：

1.改Host大法：这招对付DNS污染有用，然而GFW远没有污染DNS那么简单，除了访问Google之类的网站外没多大卵用。

2.使用代理ip地址：百度搜索“代理ip”就能找到一堆免费的国外代理服务器，不过有时间限制，速度一般很慢，而且可能有安全隐患，访问国内网站反而变慢，不灵活，应配置通用代理规则。

3.使用一键FQ软件，最早是轮子搞自由门，无借，之后有了噻风，灯笼等一键fq软件，然而这样的软件需要一直更新才能对付不断升级的GFW，且免费的软件死得更快。

4.使用付费vpn：要钱的玩意，近年来还有人发明SS加密传输对抗GFW干扰。一般速度比免费快多了，但运营商良心程度参差不齐，有些运营商开几个月后直接跑路，换另一个名字继续骗钱。

5.使用较复杂免费FQ软件，比如某g（已死）、xx-net和新出的pac代理+浏览器插件SwitchyOmega。

6.使用在线网页代理和远程桌面技术：网页代理就是在一个网页上额外显示另一个很难上去的网页，至于远程桌面，就是叫你远在国外朋友打开远程桌面，然后你控制你朋友的电脑，延迟大得飞起！

7.搬瓦工与vultr及linode三家VPS:https://www.bawagon.com/bandwagonvultrlinode/ 对应优惠码http://www.softdoz.com/linode

lantern：https://github.com/getlantern/lantern/releases

psiphon：https://s3.amazonaws.com/hum9-lwg8-qa2w/zh/index.html

Tor：https://github.com/garethflowers/tor-browser-portable/releases

SS：https://github.com/shadowsocks/shadowsocks-windows/issues/293#issuecomment-132253168 https://github.com/shadowsocks/shadowsocks/commit/5b450acfaa15cd6c2d3e8ab99f9297542df74025

SSR https://github.com/breakwa11/shadowsocks-rss

github的V2Ray 已经比Shadowsocks流行。

如果你想把socks代理转换成http代理（拦截http请求走socks代理），可以用privoxy这个东东

很多游戏里的帐号登录、搜人等操作也是TCP的，就是https请求。然而在很多游戏中，你正在开始和别人联机对战时，你走的是UDP/KCP。

**KCP**

[Github](https://github.com/skywind3000/kcp)上的KCP是纯ARQ（Automatic Repeat-request，自动重传请求）算法实现，连时钟都要外部传进来，内部不会有系统调用。本身基于UDp，如果自己实现了某种P2P、UDP协议，可以复用ARQ算法。KCP 是一个快速可靠协议，能以比 TCP 浪费10%-20%的带宽/流量的代价，换取平均延迟降低 30%-40%，且最大延迟降低三倍的传输效果，是一种双边加速办法。

**RTO翻倍vs不翻倍：**TCP超时计算是RTOx2，这样连续丢三次包就变成RTOx8了，十分恐怖，而KCP启动快速模式后不x2，只是x1.5（实验证明1.5这个值相对比较好），提高了传输速度。

**选择性重传 vs 全部重传：**TCP丢包时会全部重传从丢的那个包开始以后的数据，KCP是选择性重传，只重传真正丢失的数据包。

**快速重传：**发送端发送了1,2,3,4,5几个包，然后收到远端的ACK: 1, 3, 4, 5，当收到ACK3时，KCP知道2被跳过1次，收到ACK4时，知道2被跳过了2次，此时可以认为2号丢失，不用等超时，直接重传2号包，大大改善了丢包时的传输速度。

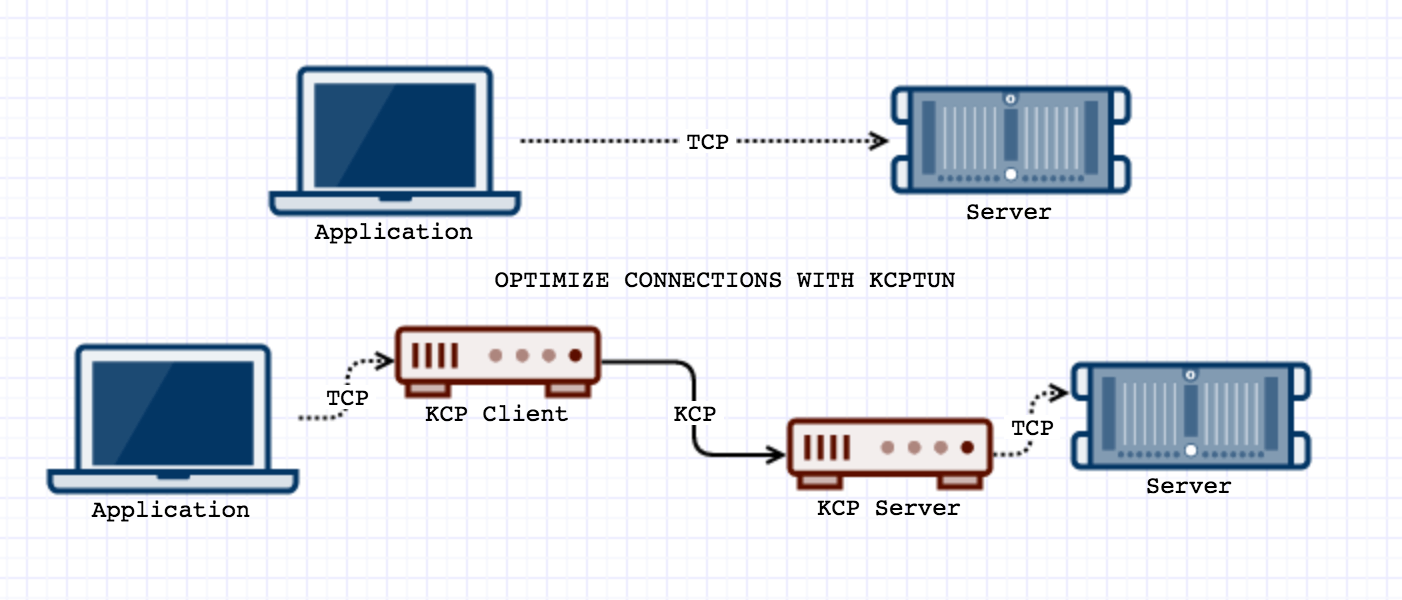
**延迟ACK vs 非延迟ACK：**TCP为了充分利用带宽，延迟发送ACK（NODELAY都没用），这样超时计算会算出较大 RTT时间，延长了丢包时的判断过程。KCP的ACK是否延迟发送可以调节。

**UNA vs ACK+UNA：**ARQ模型响应有两种，UNA（此编号前所有包已收到，如TCP）和ACK（该编号包已收到），光用UNA将导致全部重传，光用ACK则丢失成本太高，以往协议都是二选其一，而 KCP协议中，除去单独的 ACK包外，所有包都有UNA信息。

**非退让流控：**KCP正常模式同TCP一样使用公平退让法则，即发送窗口大小由：发送缓存大小、接收端剩余接收缓存大小、丢包退让及慢启动这四要素决定。但传送及时性要求很高的小数据时，可选择通过配置跳过后两步，仅用前两项来控制发送频率。以牺牲部分公平性及带宽利用率之代价，换取了开着BT都能流畅传输的效果。

https://github.com/skywind3000/kcp/blob/master/README.en.md。 代表软件kcptun实现的udp可靠传输隧道。mKCP 与 KCPTUN 同样是 KCP 协议，但两者并不兼容。使用这些小众协议，尤其要注意软件版本与实现的协议版本，极有可能存在不兼容问题。

<https://github.com/vzex/dog-tunnel> 基于kcp的p2p端口映射工具，同时支持socks5代理



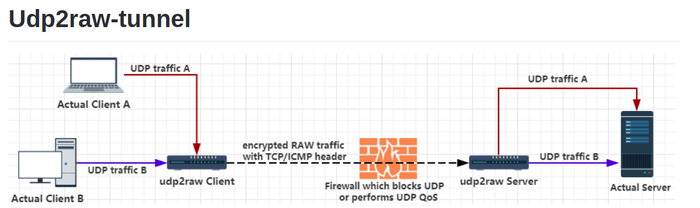
QUIC（Quick UDP Internet Connection）是谷歌制定的一种基于UDP的低时延的互联网传输层协议，对应TCP+TLS+SPDY，相比较kcp重了一些。QUIC具有SPDY（SPDY是谷歌研制的提升HTTP速度的协议，是HTTP/2.0的基础）所有的优点；0-RTT连接；减少丢包；前向纠错，减少重传时延；自适应拥塞控制， 减少重新连接；相当于TLS加密。QUIC协议内置了TLS栈，实现了自己的传输加密层。QUIC协议实现了自己的会话标记方式，称为连接UUID。当设备网络环境切换时，连接UUID不会发生变化，因此无需重新进行握手。详细介绍：<https://blog.csdn.net/dxpqxb/article/details/76821663。>

bbr Google的TCP拥塞控制算法，从 4.9 开始，Linux 内核上可以开启，兼容QUIC

不推荐使用udt

锐速 TCP加速，是基于多倍暴力发包，需要内核适配，闭源软件。他的发送包根本不遵守网络的控制规则不论这个包是否成功.网关是否接受这次传输请求.他都一股脑的发出去实际包能否到达用户你从服务器根本看不到。

Udpspeeder 暴力发包来尽量保证UDP的可靠传输，带宽换取游戏低延迟



Brook 代理搭建（需要自己有国外的ip）： <https://www.cnblogs.com/wincai/p/9166045.html>