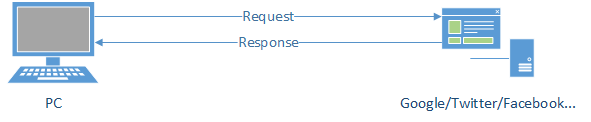
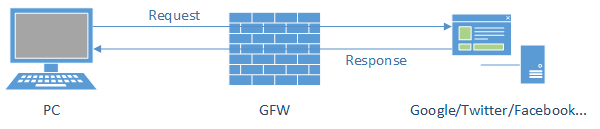
## long long ago…

在很久很久以前，我们访问各种网站都是简单而直接的，用户的请求通过互联网发送到服务提供方，服务提供方直接将信息反馈给用户 

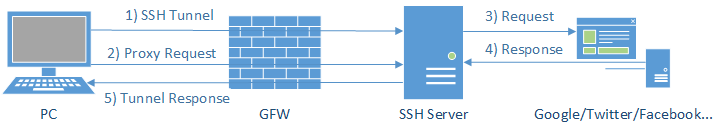
## when evil comes

然后有一天，[GFW](http://zh.wikipedia.org/wiki/%E9%87%91%E7%9B%BE%E5%B7%A5%E7%A8%8B" \t "/tmp/wps-i69/x/_blank) 就出现了，他像一个收过路费的强盗一样夹在了在用户和服务之间，每当用户需要获取信息，都经过了 GFW，GFW将它不喜欢的内容统统过滤掉，于是客户当触发 GFW 的过滤规则的时候，就会收到 Connection Reset 这样的响应内容，而无法接收到正常的内容 

## ssh tunnel

聪明的人们想到了利用境外服务器代理的方法来绕过 GFW 的过滤，其中包含了各种HTTP代理服务、Socks服务、VPN服务… 其中以 ssh tunnel 的方法比较有代表性

1) 首先用户和境外服务器基于 ssh 建立起一条加密的通道 2-3) 用户通过建立起的隧道进行代理，通过 ssh server 向真实的服务发起请求 4-5) 服务通过 ssh server，再通过创建好的隧道返回给用户



由于 ssh 本身就是基于 RSA 加密技术，所以 GFW 无法从数据传输的过程中的加密数据内容进行关键词分析，避免了被重置链接的问题，但由于创建隧道和数据传输的过程中，ssh 本身的特征是明显的，所以 GFW 一度通过分析连接的特征进行干扰，导致 ssh 存在被定向进行干扰的问题

## shadowsocks

于是 [clowwindy](https://github.com/clowwindy/shadowsocks" \t "/tmp/wps-i69/x/_blank) 同学分享并开源了他的[解决方案](http://www.v2ex.com/t/32777" \t "/tmp/wps-i69/x/_blank)

简单理解的话，shadowsocks 是将原来 ssh 创建的 Socks5 协议拆开成 server 端和 client 端，所以下面这个原理图基本上和利用 ssh tunnel 大致类似

1、6) 客户端发出的请求基于 Socks5 协议跟 ss-local 端进行通讯，由于这个 ss-local 一般是本机或路由器或局域网的其他机器，不经过 GFW，所以解决了上面被 GFW 通过特征分析进行干扰的问题 2、5) ss-local 和 ss-server 两端通过多种可选的加密方法进行通讯，经过 GFW 的时候是常规的TCP包，没有明显的特征码而且 GFW 也无法对通讯数据进行解密 3、4) ss-server 将收到的加密数据进行解密，还原原来的请求，再发送到用户需要访问的服务，获取响应原路返回 