网络代理技术原理

什么是网络代理？

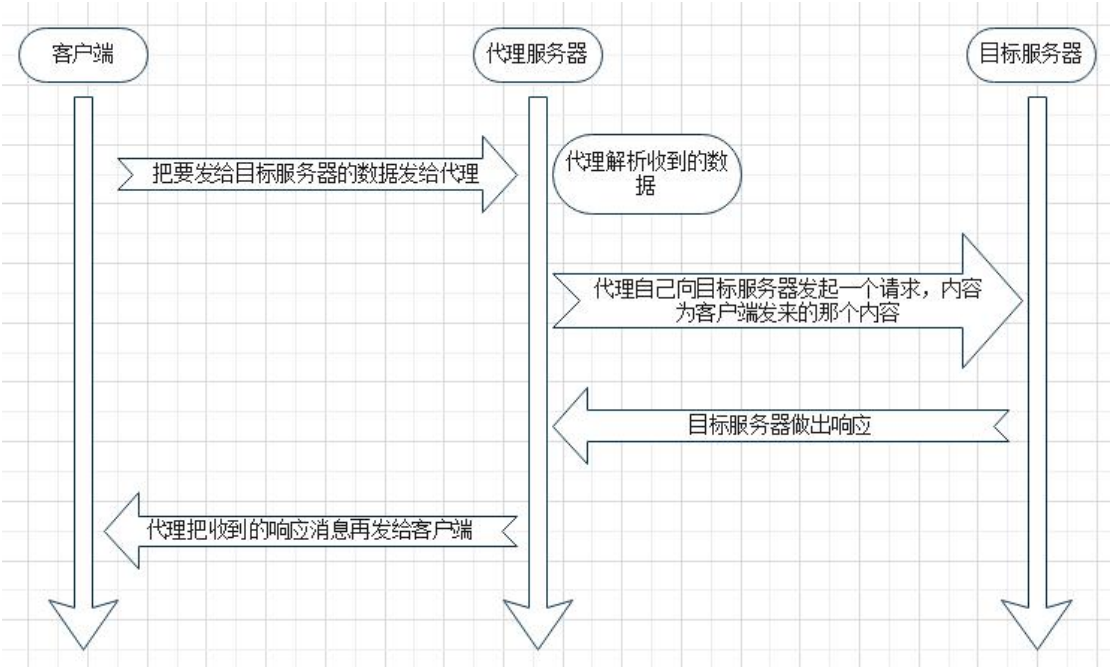
上网这种事原来就只是客户端和服务器之间的事，后来，客户端这边的上网情况变了，可能由于网管做了上网行为的限制，又或是接入运营商做了一些访问限制，导致客户端这边无法访问到某些目标服务器了。不过客户端还能访问其他未被限制的服务器，于是一个不算太难想到的想法出现了：让客户端能正常访问的那个服务器帮我们上网，我（指客户端）把要访问的数据发给它（能访问到的服务器），它再帮我去访问目标服务器，得到结果后再发回给我，我便突破了本地网管/运营商的限制，这种上网方式便是**正向代理上网**。正向代理技术不一定就是这么诞生的，但一定是为了突破上网限制而兴起的。可能是网管的限制，可能是运营商的限制，也可能是服务端对客户端源 IP 的限制。

最早的上网主要是指访问网站，使用 http 的协议，于是要代理上网的话也是基于 http 的代理，后来服务数据可能不走 http 了，用了 ssl 或仅基于 tcp，于是基于 tcp 层的 socket 代理出现，当然也有基于 udp 层的代理。代理上网技术主要是在传输层及以上的层次，有没有网络层及以下的代理呢，那个可能就不叫代理技术了，而叫 VPN 技术，本文不做这方面的讲解。正向代理技术主要有三种：HTTP 代理、HTTPS 代理、socks 代理

HTTP 正向代理：

http 协议使用的是明文，早期的网络协议都是用的明文，所以很容易遭到中间人攻击，http 的正向代理服务器就充当了这个中间人，正规的 http 服务器当然不会去随意窃取和篡改我们的信息，所以我们要使用开源的 http 正向代理服务器。http 正向代理的原理是：

在配置了 http 代理的客户端，它的浏览器在上网时，不会直接把请求的数据包发给目标服务器，而是发给了代理服务器，代理服务器收到数据包后，解析 http 头部的字段，获取客户端要访问的目标，然后代理服务器自己作为一个客户端向目标服务器发起相同的请求，获得目标服务器返回的结果后，再把数据发给最终的客户端



因为 http 协议是明文的，所以代理服务器很容易就看到了客户端发来的报文里的内容，客户端发来的 Http 报文如下（示例）

GET　http://www.test.com/ HTTP/1.1

User-Agent: xxxx

Proxy-Connection: Keep-Alive

Connection: Keep-Alive

Host: www.test.com

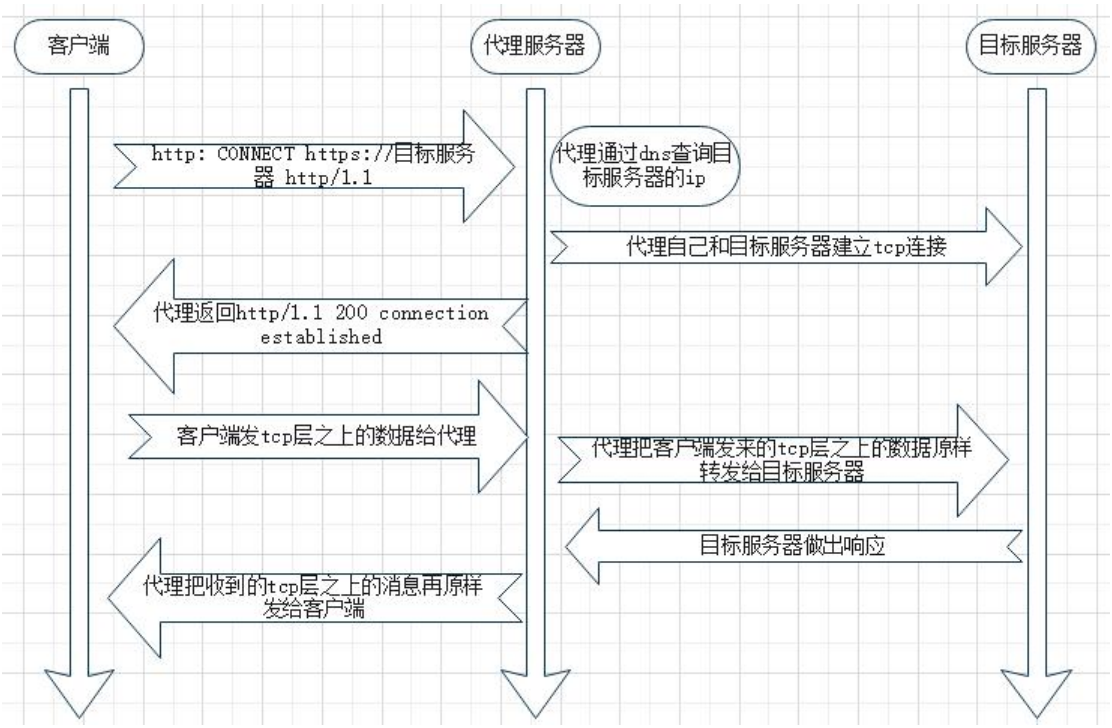
Cookie:　fsfsdfsdfsfsfds

… … …

然后代理服务器知道了客户想向 www.test.com（目标服务器）发起 get 请求，于是代理先查 www.test.com 的 Ip，再向 www.test.com 发起 tcp 连接，再发起 get 请求，目标服务器有回复后，代理再把回复的内容发给客户端。代理服务器做了一个中间人的角色。原理和实现的方法都是比较简单的，http 正向代理服务器的功能不止如此，还可以做一些过滤，如过滤目标服务器发来的广告信息和不良信息，以及阻止客户端访问某些内容。http 正向代理服务器在帮客户端上网之前可以向客户端索要验证信息，比如用户名和密码，这些信息也是放是 http 报文的头部某字段里，是明文的，客户端验证通过后才能正常通过代理上网。

HTTPS 正向代理：

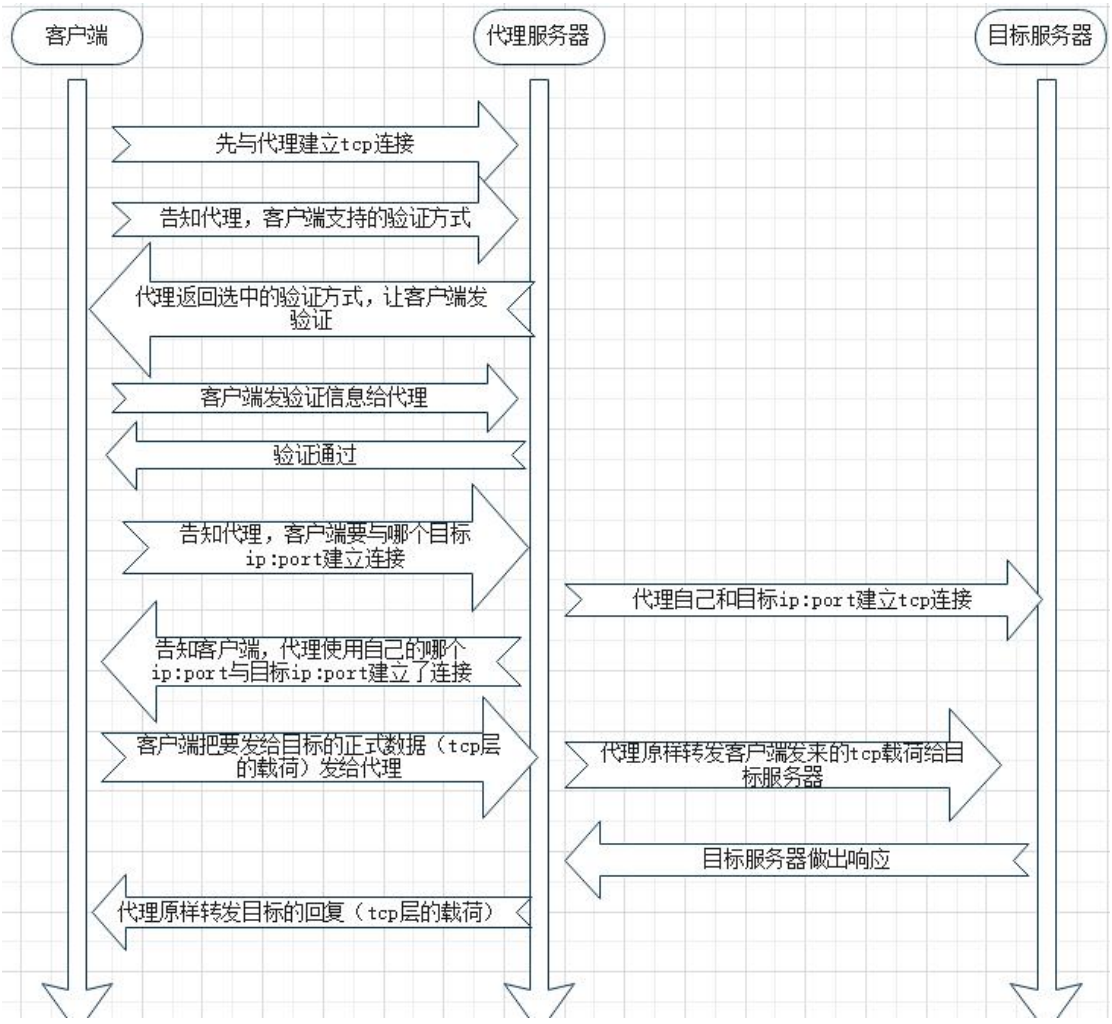
https 用到了 ssl 协议，在 ssl 层建立了安全连接后，才发起 http 请求，如果 https 代理服务器也充当中间人角色的话，它不可能有其他目标服务器的 ssl 证书及私钥，它若用自己的证书的话，也和其他目标网站不匹配，客户端浏览器就会提示连接不安全。所以一般 https 代理服务器不会做中间人角色，它只做了 tcp 层之上的数据转发，和端口转发差不多，只不过多了前面的一些事宜。



客户端首先也是发起 http 请求，请求头不再是 GET/PUT/POST 之类的，而是 CONNECT，表示希望与目标服务器建立 tcp 层的隧道连接，让代理服务器原样转发 tcp 层的载荷数据，代理收到此 http 报文后，通过 dns 查询目标服务器的 ip，再和此目标建立 tcp 连接，连接建立后，再向客户端返回 http 报文，内容为 http/1.2　200　表示连接建立了。然后客户端把正式的要向目标服务器发送的数据（如 ssl 层的交互消息等）原样发给代理，代理原样发给目标，目标返回后，代理再原样发给客户端。因为客户端和目标是通过 ssl 层来传输的 http，即使用 https 协议，所以代理这个中转站只是转发了 tcp 层的数据，无法查看它们具体发了什么。如果代理服务器需要客户端的身份验证的话，也是在前面的 Http 报文中说明，把用户名及密码放在 http 报文的头部中，验证通过再进行下面的 tcp 报文转发。

Socks5 正向代理：

socks 代理协议有多个版，常用的有 socks4、socks4a、socks5
socks4 仅支持 tcp 转发，socks4a 多了可以在代理那边做 dns 解析，socks5 多了 udp 的转发支持。socks5 支持 tcp 的转发及 udp 的转发，可以在代理那边做 dns 解析。这个 tcp 转发和 Https 代理里的那个转发一样，只是前面的协商协议不是 http 的，是 socks5 特有的，也是明文传输。



具体的交互报文就不写了，了解原理就行。

常用的 Linux 下的正向代理服务器：

代理服务器软件	代理类型	默认 port	特点
Squid	http/https	3128	带缓存的
Privoxy	http/https	8118	带过滤，可二次代理，不支持客户端身份验证
Varnish	http/https	6081	带缓存，性能比 squid 强
Polipo	http/https	8123	不过滤，可二次代理
TinyProxy	http/https	8888	可嵌入式，体积小
SS5	socks5	1080	版本旧，适用于 Centos6
Dante	socks5	1080	版本较新，可用于 Centos7

有的客户端软件本身不支持走代理通道上网，可以借助其他的代理接管器去上网，windows 下的代理接管器软件有 SocksCap32 或 SocksCap64，具体的用法这里不讲解。

前面讲的 http/https/socks5 正向代理，在与代理交互时使用的协议都是明文的，不太安全，而且由于是明文的，运营商的上网行为管理系统很容易就识别了，秒秒钟就可以把这个流量给阻断了。所以前面的那些代理服务器常用于公司内网。要想通过公网去使用以上的正向代理服务，得看代理服务器放在哪里，如果是在国内，一般可以正常访问，如果是在境外，很快就被墙了。

所以有人开发了可以跨越万里长城的正向代理服务器软件及客户端软件。原理也很简单，就是在客户端和代理服务器之间的流量使用了加密的算法进行加密，让其他人（运营商）无法识别，当然用的人多了，某些特征就比较明显了，也容易被墙。于是有了升级的版本，把代理通道的流量和正常的 https 流量混杂在一起，让上网行为管理系统也无法识别。因为外表上它就是正常的 Https 流量。这类正向代理服务有 ShadowSocks、ShadowSocksR、v2ray 等。

ShadowSocks 只是做了流量的加密，特征较明显，易被墙
ShadowSocksR 在 ShadowSocks 的基础上加入的流量的混淆，用的人多了也易被墙
v2ray 把加密的流量放在 https 里，且 v2ray 服务器可以和正常的 Https 服务共用通道，仅从技术上无法被墙，但用的人多了也会被举报。
还有其他的类似软件，是由其他商家开发的，用的协议也是上面的几个，只是服务器和客户端软件是他们自己写的，要收费的。

这里说明一下：国内的用户只能通过正常合法的通道上网，比如在电信运营商那里开通宽带/光纤上网，或使用有上网许可的手机进行 2/3/4/5G 数据接入上网，若要使用 vpn 或代理技术上网的话 得使用 有备案及相应运营许可的服务商提供的 VPN、代理服务器。个人未备案就搭建 vpn/代理服务器是非法的。

反向代理：

反向代理用到的原理和 http/https/socks 正向代理的原理是一样的。只是方向不同，两者的具体区别如下：

正向代理：是代替客户端向目标服务器发起请求，获得目标服务器的回复后，正向代理服务器再把数据发回给客户端；客户端不管要访问的目标服务器是谁，都统统把数据发给了正向代理服务器。目标服务器并不知道真实的客户端是谁，在目标服务器那里看到的直接客户端就是那个正向代理服务器。

反向代理：主要用于服务端那边，反向代理服务器通常就充当了对外的正式服务器，客户端访问某个网站时，直接访问到的那个目标服务器可能就是一个反向代理，它可以把客户的不同数据分流发给后面的不同服务器，起到负载均衡的作用以及隐藏后面的真实服务器的 ip 地址。

具体的服务器的搭建及配置不在本文档中列出了。可以参考作者的其他文档。

作者：Cof-Lee

2020-05-22