# 浅谈在代理环境中的 DNS 解析行为

2019-07-22　技术向　约 6.1 千字

虽然 Fake IP 这个概念早在 2001 年就被提出来了，但是到 Clash 提供 fake-ip 增强模式以后，依然有很多人对 Fake IP 这个概念以及其作用知之甚少。本文就简单谈谈在代理环境中，TCP 连接建立之前发生的事。由于移动设备操作系统中网络栈相对复杂，本文的例子也并不一定适用于移动端环境。文章中也许会存在很多错误，也希望各路大佬的勘误和斧正。

## 不使用代理

如果在不使用任何代理的情况下，打开一个没有命中 DNS 缓存的网站（比如 `blog.skk.moe`）的时候，浏览器和操作系统大概会执行这么一些操作：

1. 浏览器自己都有 DNS 缓存机制，因此浏览器会先开始寻找自己的缓存，不过没有找到 `blog.skk.moe` 的解析结果
2. 浏览器通过调用操作系统的 `getaddrinfo` 方法，向操作系统寻求解析结果
3. 操作系统自己也有一层 DNS 缓存，但是现在操作系统从自己的缓存中依然找不到这一结果
4. 在系统的网络设置之中有设置上游 DNS 地址，假设操作系统中设置的是 `119.29.29.29`，那么操作系统会向 `119.29.29.29` 发起解析请求（UDP 流量）拿到 `blog.skk.moe` 的 IP
5. 当然如果 `119.29.29.29` 自己没有 `blog.skk.moe` 的解析结果会找它的上游去要。不过我们不关心这一点，反正最后 `119.29.29.29` 会把 `blog.skk.moe` 的解析结果返回给设备的操作系统
6. 现在，浏览器已经可以开始向 `blog.skk.moe` 的 IP 发起 HTTPS 连接了

以上是打开一个网页常见的 DNS 解析流程，对于其它非 HTTP 的 TCP 连接（比如 SMTP）也都差不多是这个流程——由于 TCP/IP 的协议特性，在应用发起 TCP 连接时，会先发出一个 DNS question（发一个 IP Packet），获取要连接的服务器的 IP 地址，然后直接向这个 IP 地址发起连接。

## 设置代理并使用直连

现在，我们在应用程序（比如我们的浏览器、或者其它应用）中设置了代理，但是这个代理不涉及到任何远端服务器（直连模式）。接下来以设置了 SOCKS5 代理的浏览器为例。

1. 浏览器不再需要从自己的 DNS 缓存中寻找 `blog.skk.moe`，因为已经有了 SOCKS5 代理，浏览器可以直接将域名封装在 SOCKS5 流量之中发往代理客户端
2. 代理客户端从 SOCKS5 流量中抽出 `blog.skk.moe` 这个域名并设法获得解析结果
3. 代理客户端将你的 SOCKS5 流量还原成标准的 TCP 请求
4. 代理客户端将这个 TCP 连接建立起来，在这个例子之中 TCP 连接承载的是 HTTPS

之前由于获取解析结果是浏览器在操作，而大部分浏览器都会选择调用系统的 `getaddrinfo` 方法，因此如果你想要在 DNS 上做一些黑魔法就只能在操作系统层面实现，比如在本机或者别处架设一个带黑魔法的 DNS 服务器，然后你系统中设置使用这个 DNS 服务器。现在 DNS 解析是由代理客户端执行，因此在代理客户端上就可以实现一些黑魔法。比如 Surge 自己实现了一个 DNS Server 可以并发向多个上游同时发起查询、比如 V2Ray 可以实现不同域名的查询分流，等等。当然代理客户端也可以使用操作系统的 `getaddrinfo` 方法。

## 设置代理并将流量转发到远端服务器

现在在上一步的基础之上，我们为代理服务器设置了一个远端服务器，这个代理会使用 某种协议 和远端服务器通信，并且这种协议和 SOCKS5 一样支持将域名封装在传输中。浏览器和代理客户端之间依然使用 SOCKS5 通信。

1. 因为已经有了 SOCKS5 代理，浏览器可以直接将域名 `blog.skk.moe` 和整个请求封装在 SOCKS5 流量之中发往代理客户端
2. 代理客户端从 SOCKS5 流量中抽出 `blog.skk.moe` 这个域名以及其它数据
3. 代理客户端使用 某种协议 将浏览器发出的 SOCKS5 的流量重组并发给远端服务器
4. 远端服务器使用相同的 某种协议 从流量中获得其中的域名 `blog.skk.moe`
5. 远端服务器的代理服务端发起了一次 DNS 解析请求试图解析 `blog.skk.moe` 。绝大部分情况下，代理的服务端都会直接使用操作系统的 `getaddrinfo` 方法、也就是由远端服务器的操作系统负责 DNS

这一次，不论是代理客户端还是你的浏览器都没有进行 DNS 解析，DNS 解析是在远端服务器上进行的。因为 某种协议 支持封装域名，然后这一次和 `blog.skk.moe` 连接的是远端服务器，考虑到针对 CDN 优化，DNS 解析自然需要在远端服务器上执行。

现在我已经介绍了通过代理直连和通过代理发送给远端服务器了。但是毫无疑问，我相信本文所有的读者自己使用的上网方式都不会是全面直连或者全面代理。这就是接下来要讲的：

## 设置代理并使用 IP 规则和域名规则进行分流

分流是一个麻烦事。一般情况下，你可能会需要使用域名进行分流（不论是白名单还是黑名单）。不过更多情况下你会使用到基于 IP 的规则来进行分流。

先来看第一个例子：使用域名规则进行分流。

1. 浏览器将带有域名 `blog.skk.moe` 的 HTTPS 请求封装在 SOCKS5 流量之中发往代理客户端
2. 代理客户端从 SOCKS5 流量中抽取出域名 `blog.skk.moe`
3. 代理客户端开始将 `blog.skk.moe` 和域名规则列表开始比较。这个列表可以是白名单或黑名单，域名可能也没有匹配上。反正最终比较得出的结果就是 `blog.skk.moe` 是否需要走代理。
4. 如果不需要走代理，代理客户端剩下会做的事情和本文第二部分「设置代理并使用直连」就完全一样了；同理，需要走代理的话就需要进行本文第三部分的那个流程

使用域名规则分流很简单，除非 `blog.skk.moe` 最终是直连，否则代理客户端不需要进行 DNS 解析。

现在来看第二个例子：使用 IP 规则分流。

1. 浏览器将带有域名 `blog.skk.moe` 的 HTTPS 请求封装在 SOCKS5 流量之中发往代理客户端
2. 代理客户端从 SOCKS5 流量中抽取出域名 `blog.skk.moe`
3. 代理客户端得到 `blog.skk.moe` 的解析结果
4. 代理客户端开始将 `blog.skk.moe` 的解析结果和 IP 规则列表开始比较。这个列表可以是 cnlist 或者 MaxMind IP 数据库。反正最终得出的结果就是 `blog.skk.moe` 解析结果的 IP 是否需要走代理。
5. 如果不需要走代理，代理客户端剩下会做的事情和本文第二部分「设置代理并使用直连」就完全一样了；同理，需要走代理的话就需要进行本文第三部分的那个流程。

使用 IP 规则分流，前提首先你得有一个 IP 拿来比较。所以代理客户端必须先进行一次 DNS 解析。使用什么方法进行 DNS 解析并不重要，之前已经说过代理客户端甚至可以使用自己的黑魔法，而我们只需要关心最终代理客户端拿到了一个 IP 并且可以用于规则判定。

此时需要注意的是，虽然代理客户端获得了一个 IP，但是你只有在直连的时候，代理客户端可能（并且基本上都会）复用这个 IP；如果是将流量交给远程服务器，由于 某种协议 支持封装域名，因此远程服务器拿到的还是域名不是 IP、还需要进行一次解析。也就是说，远端服务器连接的 IP 与 代理客户端解析得到的 IP 毫无关系。

# 使用 redir / tun2socks 实现全局流量经过代理

在开始之前，我们先复习一下 TCP/IP 协议怎么说的——「在应用发起 TCP 连接时，会先发出一个 DNS question（发一个 IP Packet），获取要连接的服务器的 IP 地址，然后直接向这个 IP 地址发起连接」

全局流量代理可能会出现在路由器上或者 TUN/TAP 型的支持全局代理客户端上。用户不再主动为每个应用程序设置代理。此时应用程序是不会感知到代理客户端的存在，它们会正常的发起 TCP 连接，并且由于 TCP/IP 协议，在拿到 DNS 解析结果之前，连接是不能建立的。

1. 浏览器自己都有 DNS 缓存机制，因此浏览器会先开始寻找自己的缓存，不过没有找到 `blog.skk.moe` 的解析结果
2. 浏览器通过调用操作系统的 `getaddrinfo` 方法，向操作系统寻求解析结果
3. 操作系统自己也有一层 DNS 缓存，但是现在操作系统从自己的缓存中依然找不到这一结果
4. 在系统的网络设置之中有设置上游 DNS 地址。代理客户端可能会修改系统设置中的 DNS 到 127.0.0.1 或者别的 IP、也可能保留用户之前的设置，这无所谓，因为…
5. 操作系统发出的 DNS 解析请求会经过代理客户端并最终被截获
6. 代理客户端可以将这个解析请求原样发出去、或者用自己的黑魔法，总之代理客户端都会拿到一个解析结果
7. 代理客户端将这个解析结果返回回去，操作系统拿到了这个解析结果并返回给浏览器
8. 浏览器对这个解析结果的 IP 建立一个 TCP 连接并发送出去
9. 这个 TCP 连接被代理客户端截获。由于之前代理客户端进行的 DNS 解析请求这一动作，代理客户端可以找到这个只包含目标 IP 的 TCP 连接原来的目标域名
10. 如果是支持 redir 的代理客户端，那么代理客户端就会直接将域名和 TCP 连接中的其它数据封装成 某种协议 发给远端服务器；或者封装成 SOCKS5 后交给支持 SOCKS5 的代理客户端

如果代理客户端需要按照域名进行分流，一般会在第 6 步代理客户端解析出一个 IP 或者第 9 步代理客户端拿到域名以后。FancySS、KoolSS、SSTap 的流程大抵都是如此。

和应用程序直接将流量封装成 SOCKS5 大有不同，在类似于透明代理的环境下浏览器和其它应用程序是正常地发起 TCP 连接。因此除非得到一个 DNS 解析结果，否则 TCP 连接不会建立；代理客户端也会需要通过这个 DNS 查询动作，才能找到之后的 TCP 连接的域名。
你大概能够发现，浏览器、应用程序直接设置 SOCKS5 代理的话，可以不在代理客户端发起 DNS 解析请求就能将流量发送给远端服务器；而在透明代理模式下，不论是否需要 IP 规则分流都需要先进行一次 DNS 解析才能建立连接。

有没有办法能像直接设置 SOCKS5 代理一样省掉一次 DNS 解析呢？有，就是代理客户端自己不先执行查询动作，丢一个 Fake IP 回去让浏览器、应用程序立刻建立 TCP 连接：

# 在 redir / tun2socks 中使用 Fake IP

Fake IP 的定义出自 RFC3089。这个 RFC 定义了一种新的将 TCP 连接封装成 SOCKS 协议的方法。

1. 浏览器自己都有 DNS 缓存机制，因此浏览器会先开始寻找自己的缓存，不过并没有找到 `blog.skk.moe` 的解析结果

2. 浏览器通过调用操作系统的 `getaddrinfo` 方法，向操作系统寻求解析结果
3. 操作系统自己也有一层 DNS 缓存，但是现在操作系统从自己的缓存中依然找不到这一结果
4. 在系统的网络设置之中设置了一个专门的上游 DNS 地址，可能是用户手动设置的也可能是代理客户端设置的。不论如何，这个设置最终会使操作系统向代理客户端发起 DNS 请求
5. 操作系统发出的 DNS 解析请求会经过代理客户端并最终被截获
6. 代理客户端从解析请求中获得域名，从 Fake IP 池中选取一个 IP 建立映射
7. 代理客户端将这个 Fake IP 返回回去，操作系统拿到了这个 Fake IP 并返回给浏览器
8. 浏览器对 Fake IP 建立一个 TCP 连接并发送出去
9. 这个 TCP 连接被代理客户端截获。代理客户端抽取出 Fake IP 并反查出这个 TCP 连接中对应的域名
10. 有了 TCP 连接和域名，代理客户端可以轻易地将其使用 SOSCKS5 或者 某种协议 进行封装

有了 Fake IP，代理客户端无需进行 DNS 解析。最后不论是浏览器、代理客户端还是远端服务器都不会去和 Fake IP 进行连接，因为在代理客户端这里就已经完成了截获、重新封装。

即使按照域名规则分流，代理客户端都没有进行 DNS 解析的需要。只有在遇到了按照 IP 进行分流的规则时，代理客户端才需要进行一次解析拿到一个 IP 用于判断。即便如此，这个 IP 只用于分流规则的匹配，不会被用于实际的连接。

## FancySS 和 Surge / Clash 的区别

FancySS 是使用的 redir，Surge 的增强模式使用的是 Fake IP，Clash 的增强模式既有 redir-host 也有 Fake IP。首先把 FancySS 等路由器上常见的代理客户端和 Clash 的 redir-host 分为一类，Surge 的增强模式和 Clash 的 fake-ip 模式分为另一类。

路由器上常见的代理客户端一般内置了 dns2socks、dnscryp-proxy、PCap_DNSProxy 等等 DNS 方案、也支持按照一定的规则进行分流，但是都是用于答复应用程序的 DNS question 使其建立 TCP 连接的，除非直连，否则通过这些 DNS 方案拿到的解析结果的 IP 并不会被用上。
大部分路由器上的代理客户端，DNS 解析请求都是通过路由器本机发出（或转发到单一远端服务器进行解析），因此解析结果只能说「至少能用」（不一定是有 CDN 优化的，甚至有可能会有 DNS 污染），如果流量不经过代理客户端直接发往这些 IP 地址，一般也不会影响浏览器、应用程序的正常使用。因此路由器上的代理客户端可以实现通过 iptables 控制让某些端口、某些设备的流量不经过代理客户端。
而在 Fake IP 模式下，浏览器、应用程序都是对 Fake IP 发起连接，如果没有代理客户端对连接进行重新封转，那么这部分流量就不能被发往真实的目的 IP，因此所有流量都必须经过代理客户端，而根据端口、设备的分流就需要由代理客户端自己实现。

## 如果操作系统或者浏览器缓存了 DNS 解析结果

之前的透明代理的两个例子中，我们都假定浏览器和操作系统都没有缓存 DNS 解析结果。但是，如果操作系统或者应用程序缓存了 DNS 解析结果会发生什么？

如果是不使用 Fake IP 的 redir / tun2socks 情况下，由于操作系统、浏览器或者应用程序中的任何一个缓存了 DNS 解析结果，因此 TCP 连接可以直接根据缓存的解析结果的 IP 建立，代理客户端并没有预先收到对应的 DNS question。在这种情况下，代理客户端有可能直接将这个连接视为和 IP 连接而不是和域名连接，根据域名规则的分流可能就会因此失效，不过根据 IP 分流的规则没有失效。
如果为了避免域名分流规则失效，你可以设法阻止操作系统或者浏览器缓存 DNS 解析结果，这样每次建立 TCP 连接之前都会发送 DNS question 使代理客户端探测到域名。但是这意味着每次 TCP 连接建立都需要代理客户端进行一次 DNS 解析请求（当然代理客户端可以对 DNS 解析进行缓存避免出现延时激增）。

而对于 Fake IP 模式来说，由于代理客户端内存储有 Fake IP 和真实域名之间的映射表，因此即使操作系统或应用程序缓存了 Fake IP，在之后的 TCP 连接中，代理客户端收到流量后依然可以抽取出 Fake IP 反查出域名，因此不受 DNS 缓存的影响。

我在这里留几个问题给大家思考一下：

- 如果使用了 Fake IP，代理客户端不论域名是否真实存在都会返回一个 Fake IP 给浏览器，那么浏览器在试图访问一个不存在的域名时，错误信息应该是什么样的？会不会出现 DNS 解析失败的错误信息？
- 如果操作系统或者浏览器缓存了 Fake IP，但是代理客户端中 Fake IP 和域名的映射表丢失以后，会出现什么状况？可能会出现什么错误信息？

第二个问题很有趣。因为如果你找到了第二个问题的答案，你就会意识到 Clash 在 Fake IP 模式下偶发的无法上网的原因了。

## 参考资料


- HTTP 代理原理及实现（一） - 我的文章中举得都是 SOCKS5 的例子，如果想了解一下在 HTTP 代理中流量是如何被封装的，可以看看屈屈的这篇博客
- Surge 原理与实现 - Surge 开发者写的 Surge 早期版本的工作原理，可以了解一下 Surge 是怎么处理各种协议的流量的
- 漫谈各种黑科技式 DNS 技术在代理环境中的应用 - Kitsunebi 开发者写的文章，详细地介绍了在不同的 V2Ray 配置下的 DNS 行为，同时还有对移动端网络栈的一些介绍