生活中的 Wireshark

129

Wireshark 不只应用于企业级数据中心。在高度依赖于网络的现代生活中，Wireshark 也有广阔的用武之地，尤其是当前越来越流行的手机应用。这一部分介绍了如何在家中搭建一个环境来抓取手机上的 WiFi 网络包，分析了微博和微信等不同 App 的网络行为，揭露了家庭宽带如何被恶意劫持等。这一部分技术含量不一定很高，但是比较有趣。

假宽带真相

央视的【每周质量报告】做过一期关于网络的节目，叫“假宽带真相”。大意是说某些运营商的带宽远远达不到其承诺的标准，360 的测速软件也“有明显的设计缺陷”，所以测出的结果远高于真实带宽。

难得有个业内大新闻，我当然不会错过，当即就用 Wireshark 验证了一下。这一试才知道，原来网络测速包含了不少有意思的知识点，所以便写出来和大家分享。

我家当时用的是中国电信的 10M 宽带，从官网测到的结果如图 1 所示，下载速度达到 1235KB/秒，差不多是 10M 了。在不同时间段测试都是一样的结果，所以应该是可信的。

图 1

注意：中国特色的宽带服务是以下载速度为计算标准的，其实上传速度慢很多，上下行带宽严重不对等。这就是为什么会在图 1 中看到上传速度只有 2M。本文不关注这一点，所以只分析下载速度。

这个速度究竟是怎样测出来的呢？我用 Wireshark 抓了个包，且看下面的详细分析。

点击 Wireshark 的 Statistics 菜单，再点击 Conversations 选项，可以得到图 2 所示的窗口。从中可见测速过程中用到了 5 个 TCP 连接在下载。因为端口号是 80，所以应用层协议应该是 HTTP。

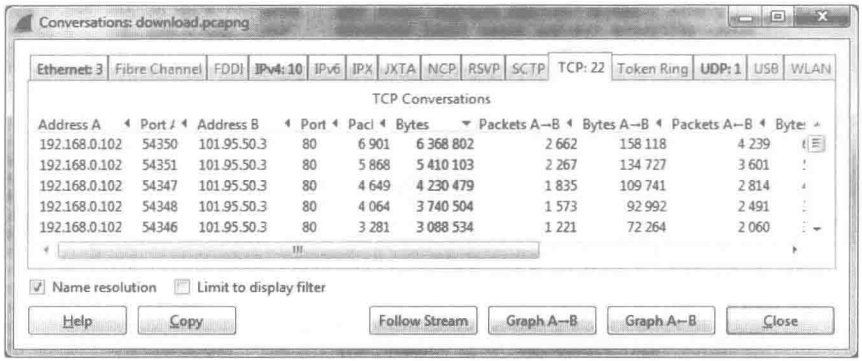


图 2

为什么要选择 5 个连接，而不是更多或者更少呢？其实连接数的选取很有讲究。之所以不用单个连接，是因为一个连接不可能时刻都在传输，有很多原因会导致它不得不短暂停滞。当某一个连接停滞时，其它的连接还可以继续传输，这样就能最大限度地利用带宽。在《固定宽带接入速率测试方法》通信行业标准中，也明确规定了“测试中使用的线程数量为 N（N≥4）”。

图 3 是其中一个连接的 Time/Sequence Number 坐标图，我是在 Wireshark 中点击 Statistics →TCP StreamGraph→Time-Sequence Graph（Stevens）菜单来生成它的。

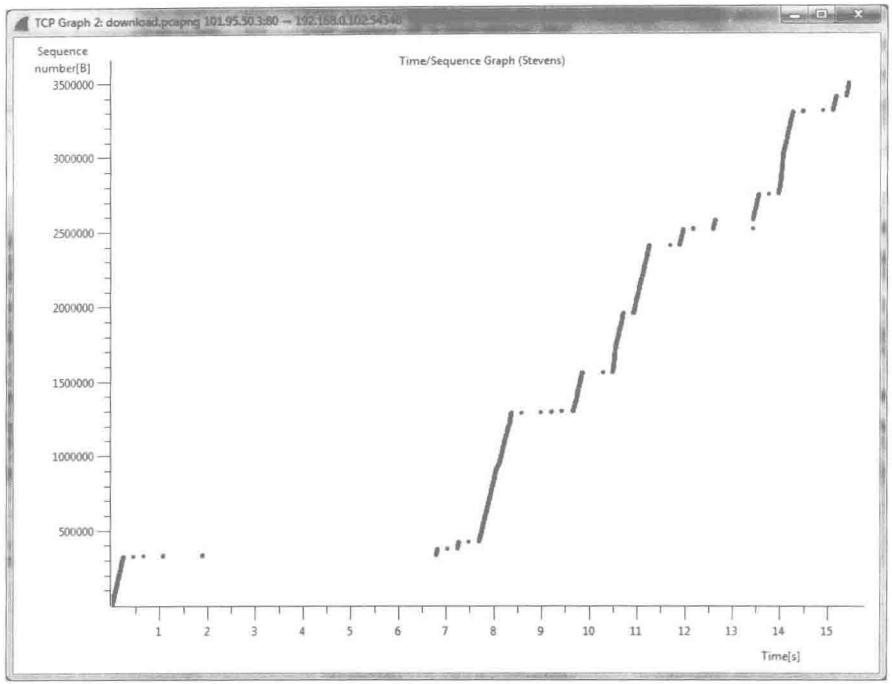


图 3

对这个连接而言，传输过程中遭遇了多次停滞，比如最严重的是 0.3～7.8 秒之间，Sequence 值几乎没有增长。还好其他 TCP 连接在这段时间里仍在正常传输，所以带宽一点都没有浪费。之所以没有用更多的连接数，是因为多到一定程度就没有意义了，甚至会影响 TCP 的拥塞控制效果。我用 iPerf 测试过的，详情可见本书的《过犹不及》一文。究竟用多少个连接数最好，这是需要测试的，估计技术人员测下来的最佳连接数是 5。随着百兆家庭带宽的普及，相信我们以后会看到更多的连接数。

再回到 Wireshark 的主界面。如图 4 所示，在下载测试开始之前，客户端是用一个 GET 方法查到下载源的，即 http://101.95.50.3/test.img。直接用 IP 的办法不错，因为不会受到 DNS 查询时间的影响。

No.	Time	Source	Destination	Protocol	Info
9	0.300174000	192.168.0.102	218.1.60.39	HTTP	GET /speed/PluginAccessLimit.do?method=acquire&uip=KD10230
10	0.304187000	218.1.60.39	192.168.0.102	TCP	[TCP Dup ACK 8#1] 80→54345 [ACK] Seq=266 Ack=1558 Win=8704
11	0.304189000	218.1.60.39	192.168.0.102	TCP	80→54345 [ACK] Seq=266 Ack=3138 Win=11520 Len=0
12	0.327770000	218.1.60.39	192.168.0.102	HTTP	HTTP/1.1 200 OK (application/json)

```
Object
  Member Key: "id"
      String value: 6585031
  Member Key: "url"
      String value: http://101.95.50.3/test.img
```

图 4

获知下载源之后，就可以建立 5 个 TCP 连接下载了。图 5 是其中的一个连接，从 Time 一栏可见响应速度相当快，GET 请求发出去 3.1 毫秒后（即 16 号包和 18 号包之间的时间差）就开始收到数据了。这是因为 101.95.50.3 位于上海电信的机房中，离我家不远。而且这应该是一台专门用来提供测速的服务器，很可能被全面优化过了。不过再怎么优化都不算作弊，电信承诺的 10M 本来就是理想状态下的带宽。看来央视曝光的假带宽问题没有发生在我身上。

No.	Time	Source	Destination	Protocol	Info
13	0.420906000	192.168.0.102	101.95.50.3	TCP	54346→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PE
14	0.424337000	101.95.50.3	192.168.0.102	TCP	80→54346 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1412
15	0.424425000	192.168.0.102	101.95.50.3	TCP	54346→80 [ACK] Seq=1 Ack=1 Win=66364 Len=0
16	0.426084000	192.168.0.102	101.95.50.3	HTTP	GET /test.img?q=0.2612285758368671 HTTP/1.1
17	0.428562000	101.95.50.3	192.168.0.102	TCP	80→54346 [ACK] Seq=1 Ack=415 Win=15744 Len=0
18	0.429200000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]
19	0.429270000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]
20	0.429299000	192.168.0.102	101.95.50.3	TCP	54346→80 [ACK] Seq=415 Ack=2825 Win=66364 Len=0
21	0.429380000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]
22	0.429520000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]
23	0.429545000	192.168.0.102	101.95.50.3	TCP	54346→80 [ACK] Seq=415 Ack=5649 Win=66364 Len=0
24	0.429638000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]
25	0.429809000	101.95.50.3	192.168.0.102	TCP	[TCP segment of a reassembled PDU]

图 5

注意：高带宽并不意味着上什么网都快。影响性能体验的因素很多，除了带宽，还有跨运营商、跨区域和服务器性能等。就算你家里有 100M 宽带，靠 VPN 连到国外网站看视频也可能很卡。

那作为第三方的 360 测速软件是否真的“有明显的设计缺陷”呢？我下载到了两个 360 测速软件，先来看第一个。如图 6 所示，测出来的带宽为 8M，略低于电信官网的宣称值。

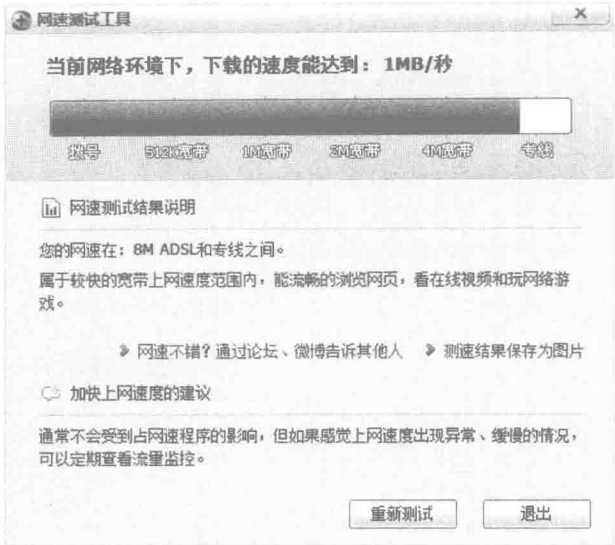


图 6

于是再用 Wireshark 分析。从图 7 可见，360 测速软件也选择了 5 个 TCP 连接来下载，端口号也是 80，与电信的方式如出一辙。原理是一模一样，差别只是服务器的响应速度，电信服务器为 3.1 毫秒，360 服务器则是 4.9 毫秒，这也许就是结果略有不同的原因。具体网络包和图 5 很像，为了不浪费篇幅我就不贴出来了。

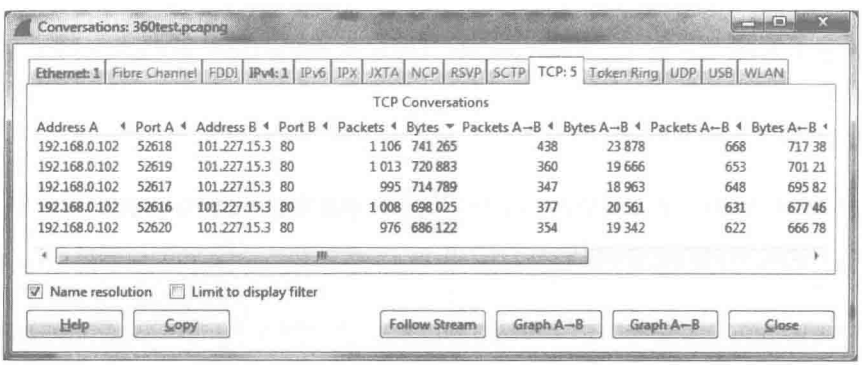


图 7

从这个工具看，360 测速软件并不存在央视所说的“明显的设计缺陷”，否则电信官网也算设计缺陷。于是我决定试一下另一个 360 测速工具。从图 8 可见，其结果接近 10M。

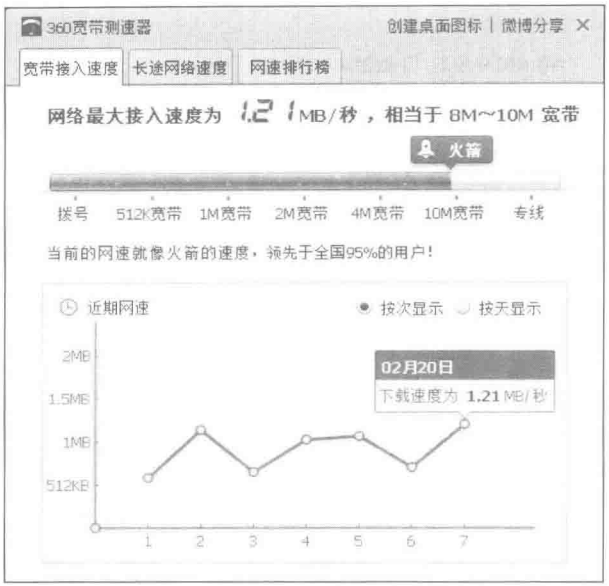
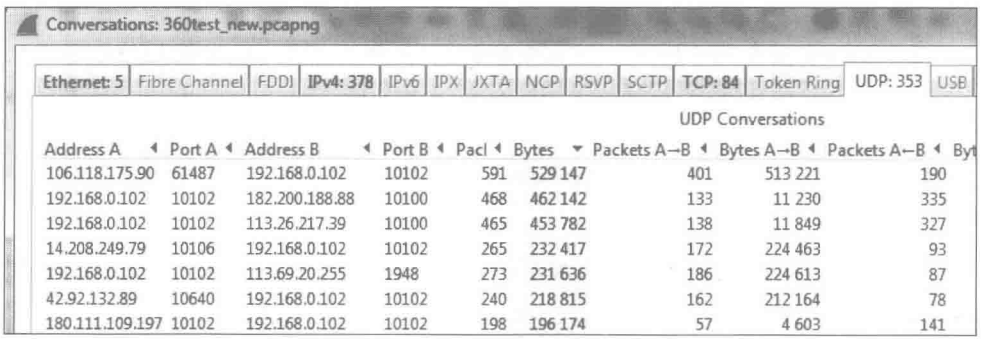


图 8

再用 Wireshark 分析。从图 9 可见，这次除了 HTTP 下载，还有不少数据是通过 P2SP 的，传输层走的是 UDP 协议。央视的专家估计也看到这个现象，所以认为这是一个设计缺陷，说“这种 P2SP 测速方法，它会去选择一些同样安装了这款软件的其他的连接节点来进行测速，只要其中有一个节点，它是在这个用户同一个小区宽带的子网里面，它的这个链路质量就非常好，网速就非常快”。我有点怀疑这个猜测，因为一个小区里有多个用户安装测速软件的概率太低了。我自己测了几次的结果都差不多，假如真有一个节点在我们小区里，应该能从图 9 的统计表中看出这个“作弊”IP 的流量。



Address A	Port A	Address B	Port B	Pacl	Bytes	Packets A→B	Bytes A→B	Packets A←B	Byt
106.118.175.90	61487	192.168.0.102	10102	591	529 147	401	513 221	190	
192.168.0.102	10102	182.200.188.88	10100	468	462 142	133	11 230	335	
192.168.0.102	10102	113.26.217.39	10100	465	453 782	138	11 849	327	
14.208.249.79	10106	192.168.0.102	10102	265	232 417	172	224 463	93	
192.168.0.102	10102	113.69.20.255	1948	273	231 636	186	224 613	87	
42.92.132.89	10640	192.168.0.102	10102	240	218 815	162	212 164	78	
180.111.109.197	10102	192.168.0.102	10102	198	196 174	57	4 603	141	

图 9

综上所述，**360 测速软件还是有节操的，它体现的是模拟现实的网速，包括 HTTP（浏览网页和刷微博之类的）和 P2SP（比如迅雷下载）。运营商提供的测速也没有作弊，不过它体现的是一个接近理想状况的网速。**那为什么央视说有些宽带不达标，但 360 测速软件却给出很高的带宽呢？我认为这不是 P2SP 导致的，而是因为这些运营商侦查到 360 正在测速，于是立即劫持，转变成在**限速点**之内测速了。如果真是这样，那也是运营商的问题，不能怪测速工具。由于我没抓到这种包，所以就不多作评论。

最后声明一下，我写这篇文章的目的不是给上海电信做广告或者为 360 正名，只是想借助这个话题演示一下 Wireshark 的应用场景。几乎所有和网络相关的问题都可以用 Wireshark 来探索学习，有时候稍微分析一下就能看得很远。我对带宽缺斤短两也不在乎，因为从来不下载电影或者美剧。网络的安全和通畅才是我最重视的，可惜这两点并不容易享受到。

手机抓包

我很久以前就想在手机上抓包了。因为随着移动 App 的流行（见图 1），手机的流量越来越大，值得研究的技术问题也会越来越多。像我这样还未融入现代社会的大叔，每个月都能用掉几百兆流量，那些摩登青年的流量之大可想而知。

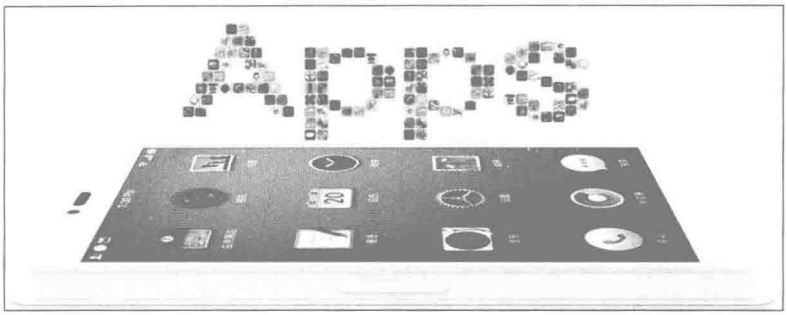


图 1

不过作为晚期拖延症患者，我迟迟没有付诸行动，直到有一天手机上的某个 App 莫名其妙地耗掉了近百兆流量，才不得不动手。我先打了个电话给中国移动，客服人员说，“我们最近收到很多例类似的报告了，原因还没有查明。”好吧，看来只能由我自己来查了，顺便搭建一个可以在手机上抓包的环境。

一番研究之后，我大概知道了业内人士都是怎么抓包的。

- 多数开发人员用 Fiddler 和 Charles 来抓，包括安卓和 iPhone。可惜它们都是针对 Web 的，不能满足我的全部需求。
- 有人说设个 HTTP 代理就可以在电脑上抓了，不过我感兴趣的协议不只是 HTTP。

- 有一些现成的安卓抓包工具，但需要 root 才能装。iOS 上的工具则没有找到。
- 搜到了一款叫 tPacketCapture 的工具，号称无需 root 也能抓，可是我的安卓测试机上不了 Google Play。

真没想到手机抓包这么麻烦，相比之下电脑抓包实在太方便了，只要装个 Wireshark 就行，分分钟搞定。那有没有办法用 Wireshark 来抓手机上的包呢？这个问题让我想起了大学寝室的网络拓扑，当时我们寝室四个人只共享一个对外的网口，所以就在我的电脑上装了两个网卡，网卡 1 对外，网卡 2 和室友们的电脑连在一个交换机上，如图 2 所示。

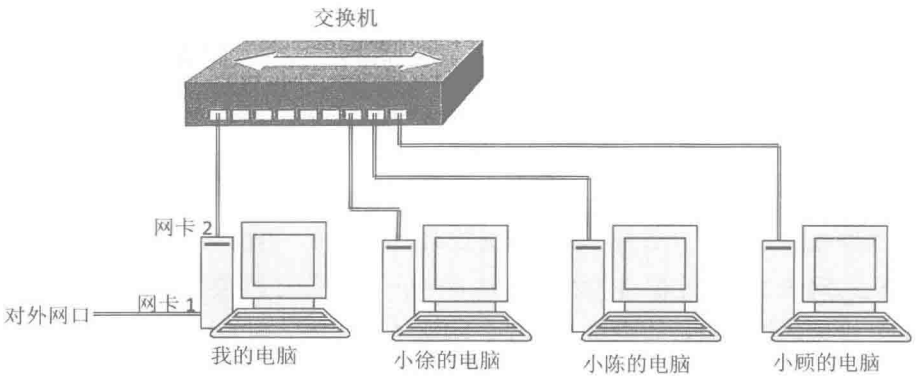


图 2

这样室友们的对外流量就是通过我的“网卡 2→网卡 1”出去了，理论上只要在我的网卡上抓包，就能知道哪位室友在和女神聊 QQ，哪位室友在祝楼主好人一生平安了（兄弟们饶命，我就说说而已，没有真抓过）。**假如把我家的网络拓扑也改成这样，让手机也通过我的电脑上网，不就可以用 Wireshark 抓到手机连 WiFi 时的包了吗？**当前我家的网络拓扑如图 3 所示，手机的网络包都通过无线路由器出去了，我得改造一下，让它们走一台已经淘汰不用的台式机。

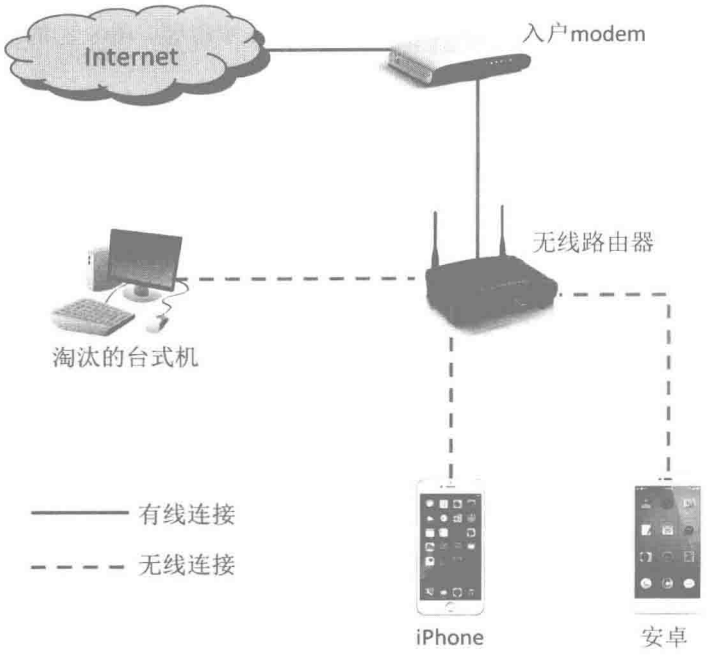


图 3

线路改造只花了几分钟：我先把无线路由器撤掉，再把入户 modem 直接连到台式机的**网卡 1** 上，这样台式机就可以上网了。接下来只要把台式机的**网卡 2**（无线的）设为热点，就能供手机上网了。网络拓扑如图 4 所示。

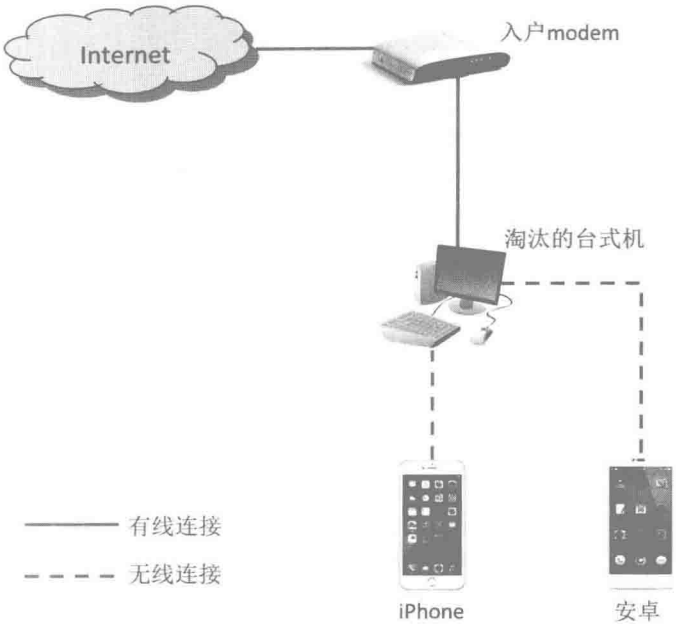


图 4

接下来的配置过程复杂一点，但也说不上很难。

1．执行图 5 的命令，将台式机的无线网络设成热点。

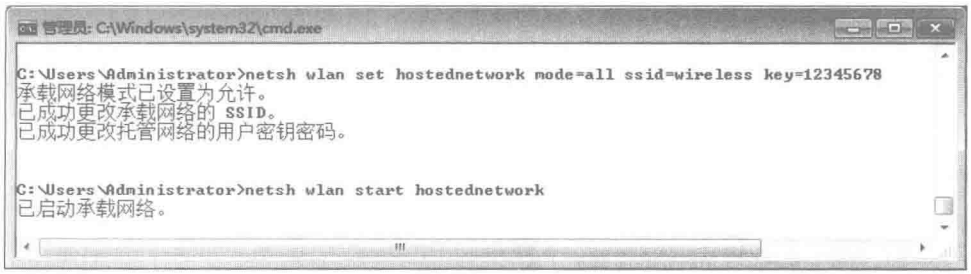


图 5

2．把台式机的网卡 1 共享给无线网络，如图 6 所示。

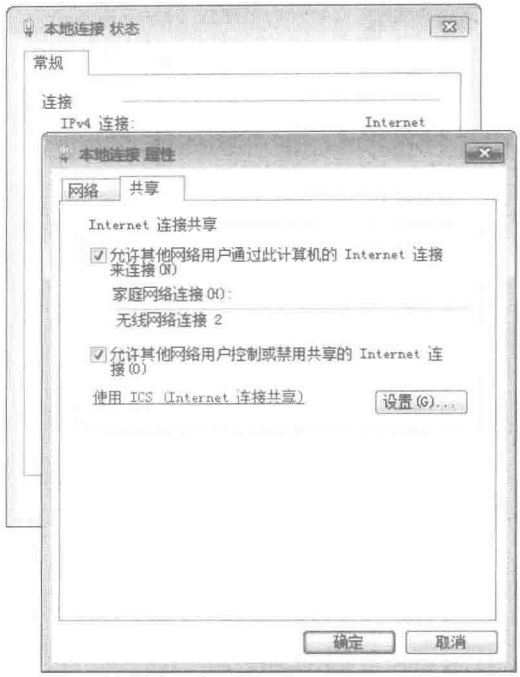


图 6

3．此时控制面板中应该可以看到 4 个连接，如图 7 所示。

图 7

4．在手机上扫到热点，输入密码就连上了，如图 8 所示。

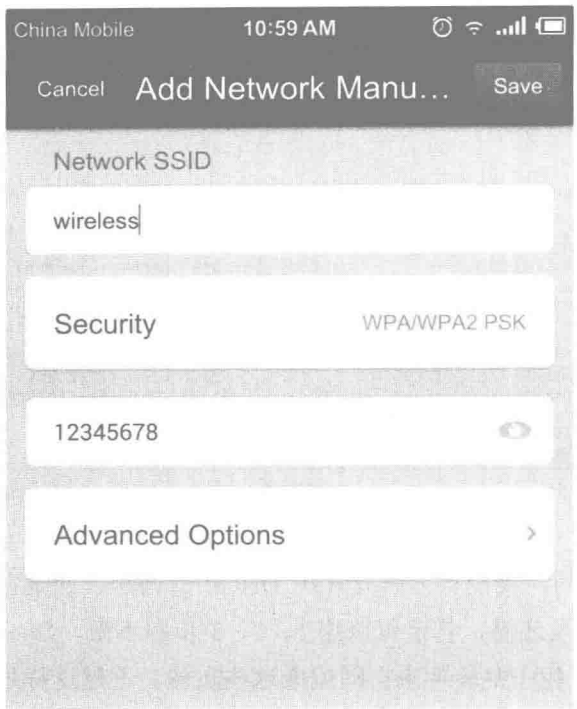


图 8

5. 在台式机上打开 Wireshark，从 Capture Interfaces 可以看到 4 个连接，勾上我们感兴趣的无线网络就可以了，本地连接没必要抓。注意有些老版本的 Wireshark 是抓不到无线网络包的，你也许需要升级到最新版本。

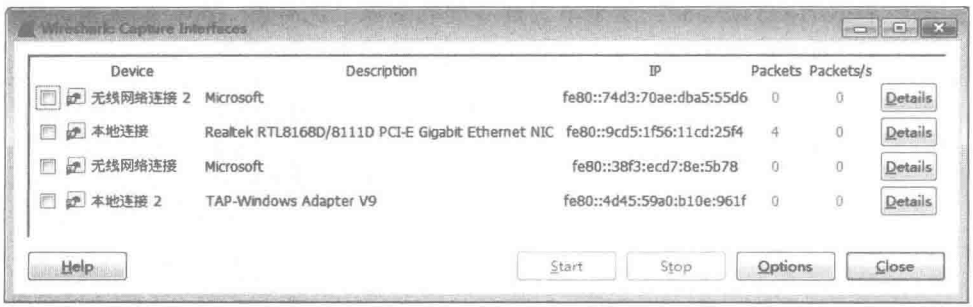


图 9

以上步骤仅供参考，因为在不同的环境中，即使严格遵循这些步骤也有失败

的可能，比如有些无线网卡天生就不支持当热点。我还遇到过一个问题，就是在第 4 步时手机一直连不上，抓包看到它发了很多 DHCP 请求给台式机，但是没有得到回复，如图 10 所示。

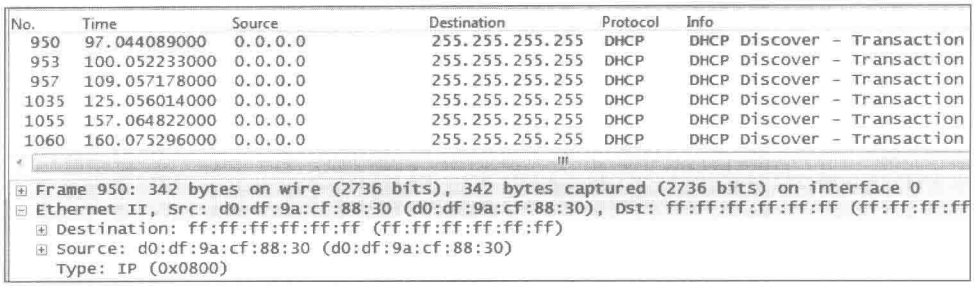


图 10

为了节省时间，我没有去研究解决 DHCP 的问题，而是在手机上人工配置了 IP。如果你比我还懒，甚至可以连 1、2、3 步都不做，Google 一下“无线网卡+WiFi 热点”，找一些软件来自动完成这些步骤。不过这样做可能遇到流氓软件，安全性不能保证。

这次网络改造非常值得，因为从此家里每个手机的网络包都可以用 Wireshark 抓到了，使我更有动力去研究手机网络。比如我想知道手机开机后的第一个网络动作是什么，抓个包就一目了然。从图 11 可见，它先通过 DNS 查询 NTP 服务器的 IP 地址，然后就发 NTP 包去同步时间了。这就是为什么手机时间用不着调整，但是走得比江诗丹顿还准。至于本文开头提到的那个耗流量 App，原来是因为它不停地刷某个网页，估计是中了木马，被我删掉了。

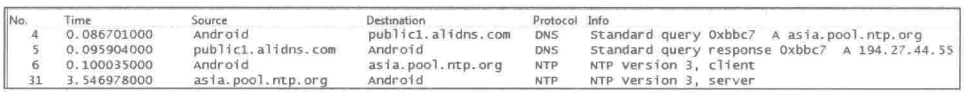


图 11

微博为什么会卡

不知道为什么，我的微博有时候会很卡，比如刷新时会一直 Loading（见图 1）。这不只是我的个人感受，很多网友都抱怨过。而装在同一个手机上的微信，连的也是同一个 WiFi，却没有这个症状。虽然这个问题出现的并不频繁，但假如我是微博的开发人员，肯定要把原因找出来。

图 1

当我的手机抓包环境搭好时，第一个想解决的问题就是这个。我随意发了一条微博，虽然没有碰到卡顿，但还是把包抓下来了。开头几个网络包如图 2 所示。

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	Android	DNS_Server	DNS	Standard query 0x917c A api.weibo.cn
3	0.009664000	DNS_Server	Android	DNS	Standard query response 0x917c CNAME weibo.cn
4	0.011417000	Android	weibo.cn	TCP	48658→80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 S
7	0.044534000	weibo.cn	Android	TCP	80→48658 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
8	0.045467000	Android	weibo.cn	TCP	48658→80 [ACK] Seq=1 Ack=1 Win=14720 Len=0

图 2

我又发了一条测试私信，可惜也没有卡顿。开头几个网络包如图 3 所示。

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	Android	DNS_Server	DNS	Standard query 0x860b A ps.im.weibo.cn
2	0.009506000	DNS_Server	Android	DNS	Standard query response 0x860b A 180.149.134.252
3	0.011703000	Android	ps.im.weibo.cn	TCP	42555→8080 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 S
4	0.040482000	ps.im.weibo.cn	Android	TCP	8080→42555 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
5	0.041463000	Android	ps.im.weibo.cn	TCP	42555→8080 [ACK] Seq=1 Ack=1 Win=14720 Len=0

图 3

虽然两次都没有重现问题，但是从网络包可见，微博的工作方式严重依赖 DNS。它在调用任何功能之前都要先向 DNS 服务器查询，得到提供该功能的服务器 IP，然后再建立 TCP 连接。最神奇的是它不会缓存查询结果，所以需要频繁地

重复查询 DNS。我才抓了两分钟包，竟然就看到了上百个查询，这会不会就是微博卡顿的原因呢？我又抓了一个发微信的包作对比，如图 4 所示。

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	Android	14.17.52.137	TCP	37613→80 [SYN] Seq=0 Win=14000 Len=0 M
2	0.033016000	14.17.52.137	Android	TCP	80→37613 [SYN, ACK] Seq=0 Ack=1 Win=14
3	0.034825000	Android	14.17.52.137	TCP	37613→80 [ACK] Seq=1 Ack=1 Win=1792000

图 4

果然，微信客户端直接就和一个 IP 地址建立了连接。不管这个 IP 是写在配置文件中的，还是之前就存在手机的缓存里的，这至少说明了微信不像微博那样依赖 DNS。

为了进一步验证这个猜测，我故意把手机上的 DNS 服务器配成一个不存在的地址。不出所料，微信还是能照常工作，但微博就再也刷不出来了。之前我手机上配的 DNS 服务器位于美国，可能有时候跨国连接不稳定，所以导致了微博的卡顿现象。考虑到这一点，我尝试配了一个国内的 DNS（见图 5），果然从此再也没卡过了，刷起来异常流畅。

图 5

当你看到这篇文章的时候，也许这个问题已经被新浪解决了，因为我已经向微博的技术人员反馈过（或者他们早已经知道）。相信解决起来也不复杂，只要像微信一样缓存 IP 就可以了。据我所知，苹果的 App Store 和小米电视也遭遇过 DNS 导致的性能问题，所以**相信还有很多设备或者程序可以利用 Wireshark 来优化，只要把使用过程的包都抓下来，说不定就能发现值得改进的地方。**

最后再补充一个小发现。我发的微博内容是“capture test, will delete it soon.”，

分享范围设成“仅自己可见”。没想到在 Wireshark 上直接就看到了明文（见图 6 底部），发私信就没有这个问题。因此我们连公共 WiFi 发微博的时候，还是要小心一点。不要以为设成“分组可见”或者“仅自己可见”就够私密了，其实在 Wireshark 上都能看到。

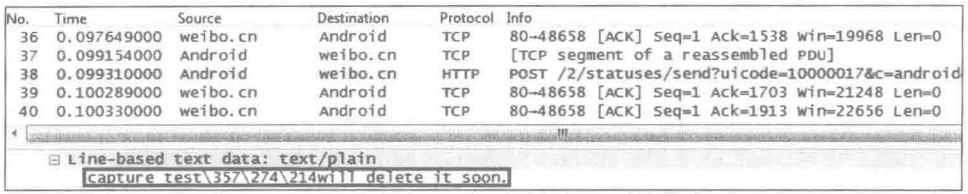


图 6

寻找 HttpDNS

这几年互联网行业有多火？假如有块陨石掉进创业园区，说不定能砸到两位互联网架构师；要是没学会几句互联网黑话，你都不好意思说自己是搞 IT 的。不久前就有位架构师在技术群里讨论鹅厂（黑话，即腾讯公司）的 HttpDNS，令我自惭形秽，因为这个词我从来没有听说过。

为了掩饰自己的孤陋寡闻，我悄悄做了点功课，发现这技术还挺有趣的。而要学习它，就得从最传统的 DNS 开始说起。

我们都知道上网的时候需要先把域名解析成 IP 地址，比如我在浏览器中输入 www.qq.com 再按回车，就会通过 DNS 查询到该域名所对应的 IP，然后再与之建立连接。但是很多人并不知道，DNS 的解析结果是很智能的。对于同一个域名，上海电信的用户一般会解析到属于上海电信的 IP 地址；北京联通的用户一般会解析到属于北京联通的 IP 地址。请看下面两个关于 www.qq.com 的不同解析结果。

上海电信用户解析到了 101.226.129.158（见图 1）。经查证，该 IP 属于上海电信[①]。

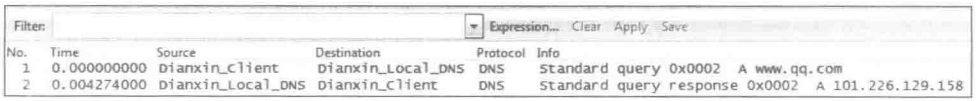


图 1

北京联通用户解析到了 61.135.157.156（见图 2）。经查证，该 IP 属于北京联通。

[①] 有很多网站可以查询 IP 地址的地理位置，本文采用的信息源是 www.ipip.net。

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	Liantong_Client	Liantong_Local_DNS	DNS	Standard query 0x0002 A www.qq.com
2	0.033494000	Liantong_Local_DNS	Liantong_Client	DNS	Standard query response 0x0002 A 61.135.157.156

图 2

这个智能技术是怎样实现的呢？原来 **DNS 支持 GSLB（Global Server Load Balance，全局负载均衡），能根据 DNS 请求所包含的源地址返回最佳结果，从而匹配同地区、同运营商的 IP，使用户体验到最好的性能。**图 3 演示了这个解析过程。

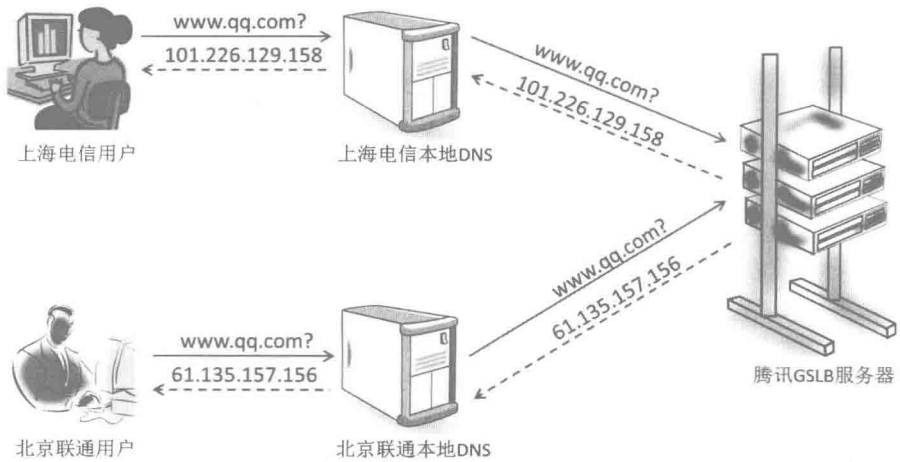


图 3

不过这个机制并非完美，比如当用户自己配错 DNS 服务器的时候就可能出问题。**图 3 的腾讯 GSLB 服务器其实并不是通过用户的地址来判断该返回什么 IP 的，而是根据 DNS 服务器的地址来判断的。假如上海电信用户偏偏要配一个北京联通的 DNS 地址，那它发送 DNS 查询时，就是由北京联通转给 GSLB 的，因此会解析到属于北京联通的 IP 地址。**由于中国的跨运营商网络一向是瓶颈，所以用户体验会很糟糕。还有些用户配的是在美国的 DNS 服务器 8.8.8.8，那就可能解析到一个位于美国的 IP 地址（启用了谷歌扩展协议的客户端除外），网速就更差了。根据公众号“鹅厂网事”的说法，他们遭遇的 GSLB 问题还有很多，比如劫持什么的，本文就不一一列举了。

那鹅厂的解决方式是什么呢？就是本文开头提到的 HttpDNS。**它允许手机上的 App 直接查询腾讯自家的 HttpDNS 服务器，因此能根据用户的地址来判断应该返回什么 IP，从而跳过传统 DNS 的影响。**换句话说，就是腾讯觉得用传统 DNS 不靠谱，所以自己做了一套解析方式，只不过这套方式是走 HTTP 协议的，图 4 演示了这个过程。

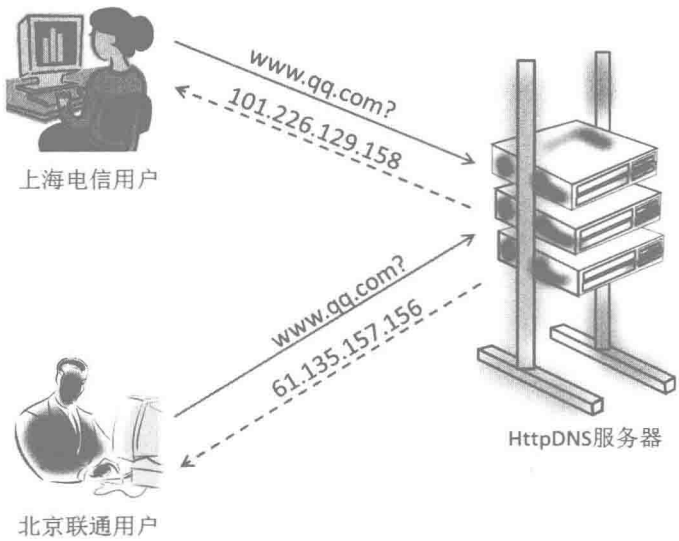


图 4

从原理上看，HttpDNS 是科学的，不过得多花些钱去部署。根据“鹅厂网事”的宣传，似乎在内部已经推广了：

“HttpDNS 已在腾讯内部接入了多个业务，覆盖数亿用户，并已持续稳定运行超过一年时间。而接入了 HttpDNS 的业务在用户访问体验方面都有了非常大的提升……国内最大的 public DNS 服务商 114DNS 在受到腾讯 DNS 的启发下，也推出了 HttpDNS 服务……”

这个宣传听上去非常吸引人。去年发生过一次全国性的 DNS 瘫痪，当时鹅厂的几个应用（比如 QQ 和微信）都能正常使用，似乎就是一个有力的佐证。连鹅厂的竞争对手似乎也加入了宣传，比如淘宝的官方微博发过一条消息，称“手机淘宝使用专为移动设计的方案”不会受到 DNS 瘫痪的影响。当时也有圈内牛人出

来解释，说这意味着手淘也开始用上 HttpDNS 了。总而言之，如果你对技术圈的八卦消息感兴趣，一定会觉得 HttpDNS 已经快颠覆 DNS 了。

事实真的如此吗？我一直有些怀疑，理由如下。

- 电脑上很多应用程序也依赖 DNS 解析，而且在电脑上很容易配错 DNS 服务器，理论上出问题的概率更大，为什么就不部署 HttpDNS？而手机上的 DNS 地址一般是运营商自动分配的，出问题的概率小，为什么反而要部署？
- 即便运营商分配的 DNS 有问题，那也可以通过行政手段来解决，何必要为此大动干戈呢？
- 国外的互联网公司也会遇到这类问题，为什么它们就没有采用 HttpDNS？
- 传统 DNS 基于 UDP 查询的速度很快，而 HttpDNS 肯定是基于 TCP 的，那还会浪费 3 次握手和 4 次挥手的时间。
- HttpDNS 如果不加密，那也很容易被劫持；如果加密了，解析效率又会大受影响。

总之有太多难以解释的疑问了，我越琢磨就越想看看 HttpDNS 的庐山真面目。为了解开这个谜题，我在搭好手机抓包环境之后，就设计了一系列实验来寻找这个传说中的新技术。

实验 1

1. 启动 Wireshark 抓包。
2. 登录手机淘宝。
3. 停止抓包并分析。

Wireshark 截屏见图 5。这个结果令我大失所望，原来手淘老老实实地用传统 DNS 查询到了服务器 d.taobaocdn.com 的 IP 地址，然后就三次握手了。也就是说，它并没有用到 HttpDNS。

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	Android	DNS	DNS	Standard query 0x6123 A d.taobaocdn.com
2	0.009048000	DNS	Android	DNS	Standard query response 0x6123 CNAME d.taobaocdn.com.
3	0.010750000	Android	d.taobaocdn.com	TCP	45134-80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PE
4	0.016174000	d.taobaocdn.com	Android	TCP	80-45134 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1
5	0.017097000	Android	d.taobaocdn.com	TCP	45134-80 [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=1205

图 5

那淘宝官方微博宣称的“专为移动设计的方案”是什么呢？我又做了个实验。

实验 2

1．登录手机淘宝。

2．然后故意把手机上的 DNS 服务器改错，发现手淘还能用。

3．退出手淘，再次登录，就再也登不上去了。

可见这所谓的方案只不过是在登录之后缓存 IP 而已，并不是用 HttpDNS 取代 DNS。既然手淘不行，我决定在手机上装个鹅厂的 QQ 试试。这一次我故意从一开始就配错 DNS。

实验 3

1．在手机上配个无效的 DNS，然后开始 Wireshark 抓包。

2．登录手机 QQ。

3．停止抓包并分析。

Wireshark 截图如图 6 所示。手机发了几个传统的 DNS 查询都没有得到响应，然后竟然就和 IP 地址 113.108.90.53 三次握手了。这个 IP 从何而来？应该就是来自 HttpDNS 了吧？然而在 Wireshark 中用尽各种 Filter 和 Find 都找不到相应的包。难不成这个 IP 是安装 QQ 时就存在配置文件中的？我从 IP 库中查到它位于 1500 公里外的深圳市，应该不会是高度智能的 HttpDNS 解析出来的。

No.	Time	Source	Destination	Protocol	Info
231	41.944338000	Android	Wrong_DNS	DNS	Standard query 0xf3cf A msfwifi.3g.qq.com
232	41.944542000	Android	Wrong_DNS	DNS	Standard query 0xe9df A configsvr.msf.3g.qq.com
233	42.463699000	Android	Wrong_DNS	DNS	Standard query 0x3e39 A monitor.uu.qq.com
237	45.945841000	Android	Wrong_DNS	DNS	Standard query 0xbb54 A strategy.beacon.qq.com
238	46.943161000	Android	Wrong_DNS	DNS	Standard query 0xf403 A monitor.uu.qq.com
239	46.969910000	Android	113.108.90.53	TCP	34777→8080 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1
240	47.010944000	113.108.90.53	Android	TCP	8080→34777 [SYN, ACK] Seq=0 Ack=1 Win=5400 Len=0 MSS=1350 S
241	47.012451000	Android	113.108.90.53	TCP	34777→8080 [ACK] Seq=1 Ack=1 Win=14720 Len=0

图 6

三个实验结果都和预想的不同，真令人心情复杂，难道技术圈的传闻并不可靠？反正时间都花了这么多了，我索性再做一个实验，彻底搞清楚 QQ 的工作方式。

实验 4

1．在手机上配一个正确的 DNS，然后开始 Wireshark 抓包。

2．登录手机 QQ。

3．停止抓包再分析。

Wireshark 截图如图 7 所示。QQ 老老实实地用传统 DNS 查到 IP，然后就三次握手了。可见它首选的就是传统 DNS，只有当 DNS 查询失败，它才直接用（可能存在配置文件里的）IP 来登录，根本没有用到 HttpDNS。

No.	Time	Source	Destination	Protocol	Info
10	0.978862000	Android	DNS	DNS	Standard query 0xf044 A msfwifi.3g.qq.com
11	0.989012000	DNS	Android	DNS	Standard query response 0xf044 A 113.108.16.66
12	0.990810000	Android	msfwifi.3g.qq.com	TCP	40188→8080 [SYN] Seq=0 Win=14600 Len=0 MSS=1460
26	1.030616000	msfwifi.3g.qq.com	Android	TCP	8080→40188 [SYN, ACK] Seq=0 Ack=1 Win=5400 Len=0
27	1.031460000	Android	msfwifi.3g.qq.com	TCP	40188→8080 [ACK] Seq=1 Ack=1 Win=14720 Len=0

图 7

一系列实验做下来，我竟然没有找到传说中的 HttpDNS。鹅厂宣传的“多个业务”究竟指的是哪些，我也不得而知。不过既然连 QQ 和手淘都没在用，我怀疑世界上本来就没多少知名 App 在用它。即便有，我也没有动力再去寻找了。当然做了这些实验也不是一无所获，至少理清楚了几个知名 App 在域名解析上的行为差异。

- **新浪微博：**一旦出现 DNS 问题就不能用，无论是否已经登录，因为它不缓存 IP。详细实验过程请看前一篇。

- **手机淘宝**：一旦出现 DNS 问题就无法登录，但是登录后再出 DNS 问题就不怕了，因为它有缓存 IP。
- **手机 QQ**：出现 DNS 问题时也能用，因为它可以直接用（可能存在配置文件里的）IP，因此受 DNS 瘫痪的影响最小。

注意：我只是描述了当前观察到的现象，并不是说某个 App 比其他的更先进。而且互联网界变化很快，说不定等你看到这篇文章时，这些 App 的行为又有所不同，甚至真的用上 HttpDNS 了，到时候抓包才知道。

这件事也促使我重新审视技术圈的信息传播。有段子说，**“美国研究机构发现，人们很容易对‘美国研究机构发现’开头的报道信以为真。”** 同样地，当 IT 大厂慷慨地分享一项技术时，当圈内大牛热情地跟着传播时，我们就会本能地觉得高大上起来。而真实情况如何，却只有自己做实验才知道。小马过河，方知深浅。