全面了解移动端DNS域名劫持等杂症：原理、根源、HttpDNS解决方案等-网络编程/专项技术区 - 即时通讯开发者社区!

本文引用了腾讯工程师廖伟健发表于“鹅厂网事”公众号上的《【鹅厂网事】全局精确流量调度新思路-HttpDNS服务详解》一文部分内容，感谢原作者的分享。

1、引言

对于互联网，域名是访问的第一跳，而这一跳很多时候会“失足”（尤其是移动端网络），导致访问错误内容、失败连接等，让用户在互联网上畅游的爽快瞬间消失。

而对于这关键的第一跳，包括鹅厂在内的国内互联网大厂，都在持续深入地研究和思考对策，本文将就鹅厂团队在这一块的技术实践，做一个深度的总结和技术分享，希

望给大家带来些许启发。

另外：即时通讯网整理的《美图App的移动端DNS优化实践：HTTPS请求耗时减小近半》、《百度APP移动端网络深度优化实践分享(一)：DNS优化篇》，也值得一读。

2、相关文章

《TCP/IP详解 卷1：协议 - 第14章 DNS:域名系统》
《美图App的移动端DNS优化实践：HTTPS请求耗时减小近半》
《百度APP移动端网络深度优化实践分享(一)：DNS优化篇》
《网络编程懒人入门(一)：快速理解网络通信协议（上篇）》
《网络编程懒人入门(二)：快速理解网络通信协议（下篇）》
《网络编程懒人入门(六)：史上最通俗的集线器、交换机、路由器功能原理入门》
《网络编程懒人入门(七)：深入浅出，全面理解HTTP协议》
《网络编程懒人入门(九)：通俗讲解，有了IP地址，为何还要用MAC地址？》
《技术扫盲：新一代基于UDP的低延时网络传输层协议——QUIC详解》
《现代移动端网络短连接的优化手段总结：请求速度、弱网适应、安全保障》
《移动端IM开发者必读(一)：通俗易懂，理解移动网络的

“弱”和“慢”》
《移动端IM开发者必读(二)：史上最全移动弱网络优化方法总结》
《脑残式网络编程入门(五)：每天都在用的Ping命令，它到底是什么？》
《脑残式网络编程入门(六)：什么是公网IP和内网IP？NAT转换又是什么鬼？》

3、正文概述

但凡使用域名来给用户提供服务的互联网企业，都或多或少地无法避免在有中国特色的互联网环境中遭遇到各种域名被缓存、用户跨网访问缓慢等问题。那么对于腾讯这样的域名数量在10万级别的互联网公司来讲，域名解析异常的情况到底有多严重呢?

每天腾讯的分布式域名解析监测系统在不停地对全国所有的重点LocalDNS（指运营商的DNS服务）进行探测，腾讯域名在全国各地的日解析异常量是已经超过了80万条（这方面，来自移动端的异常尤为突出）。这给腾讯的业务带来了巨大的损失。为此腾讯建立了专业的团队与各个运营商进行了深度沟通，但是由于各种原因，处理效率及效果均不能达到腾讯各业务部门的需求。

除了和运营商进行沟通，有没有一种技术上的方案，能从根源上解决域名解析异常及用户访问跨网的问题呢？这是包括腾讯在内的很多国内互联网大厂技术团队一直在思考的问题。

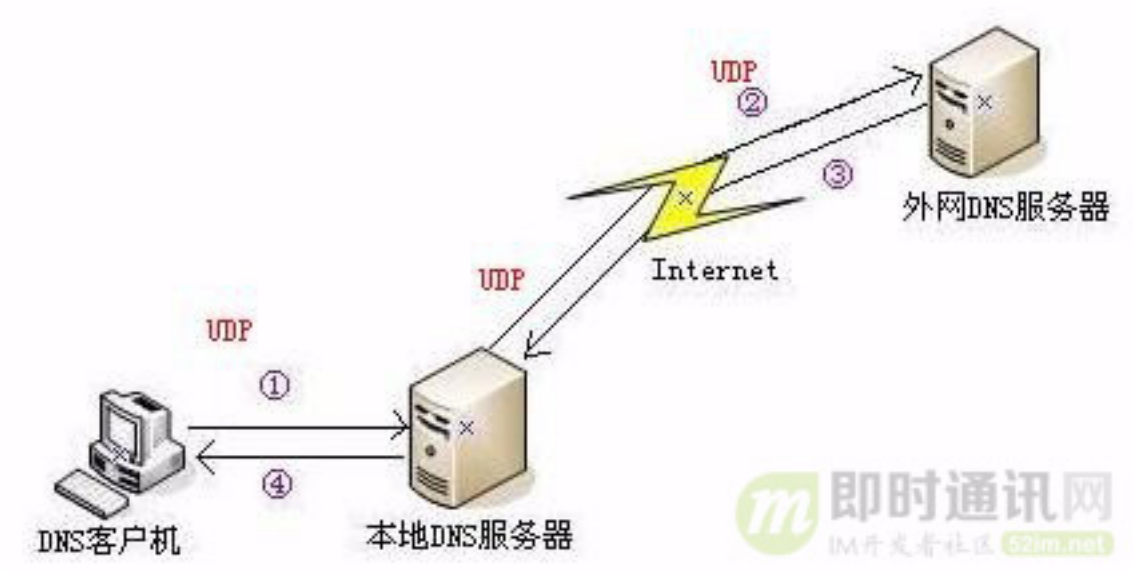
4、首先，什么是DNS?

要想理解本文将要讨论的DNS各种问题，我们需要首先来复习一下DNS的基本原理和相关知识。

4.1DNS的工作原理

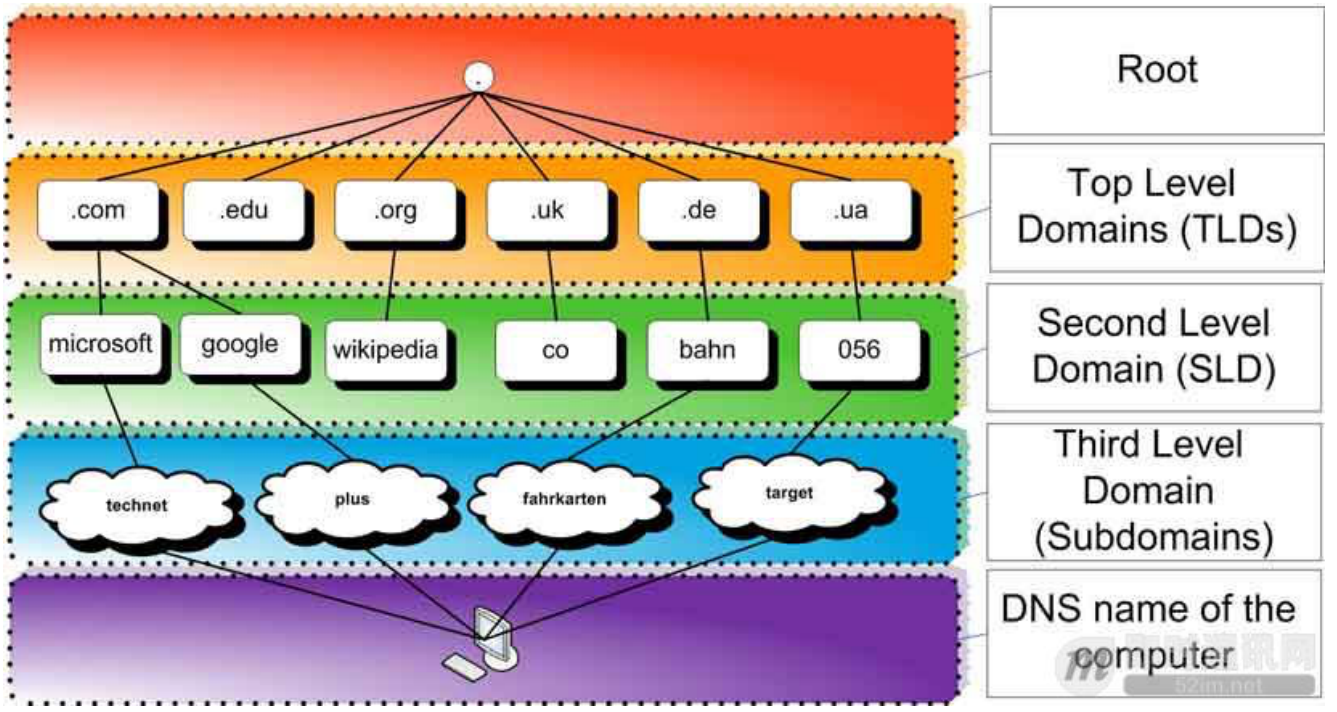
DNS（Domain Name System，域名系统），DNS 服务用于在网络请求时，将域名转为 IP 地址。能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的 IP 数串。

DNS的基一原理如下图所示：



传统的基于 UDP 协议的公共 DNS 服务极易发生 DNS 劫持，从而造成安全问题。

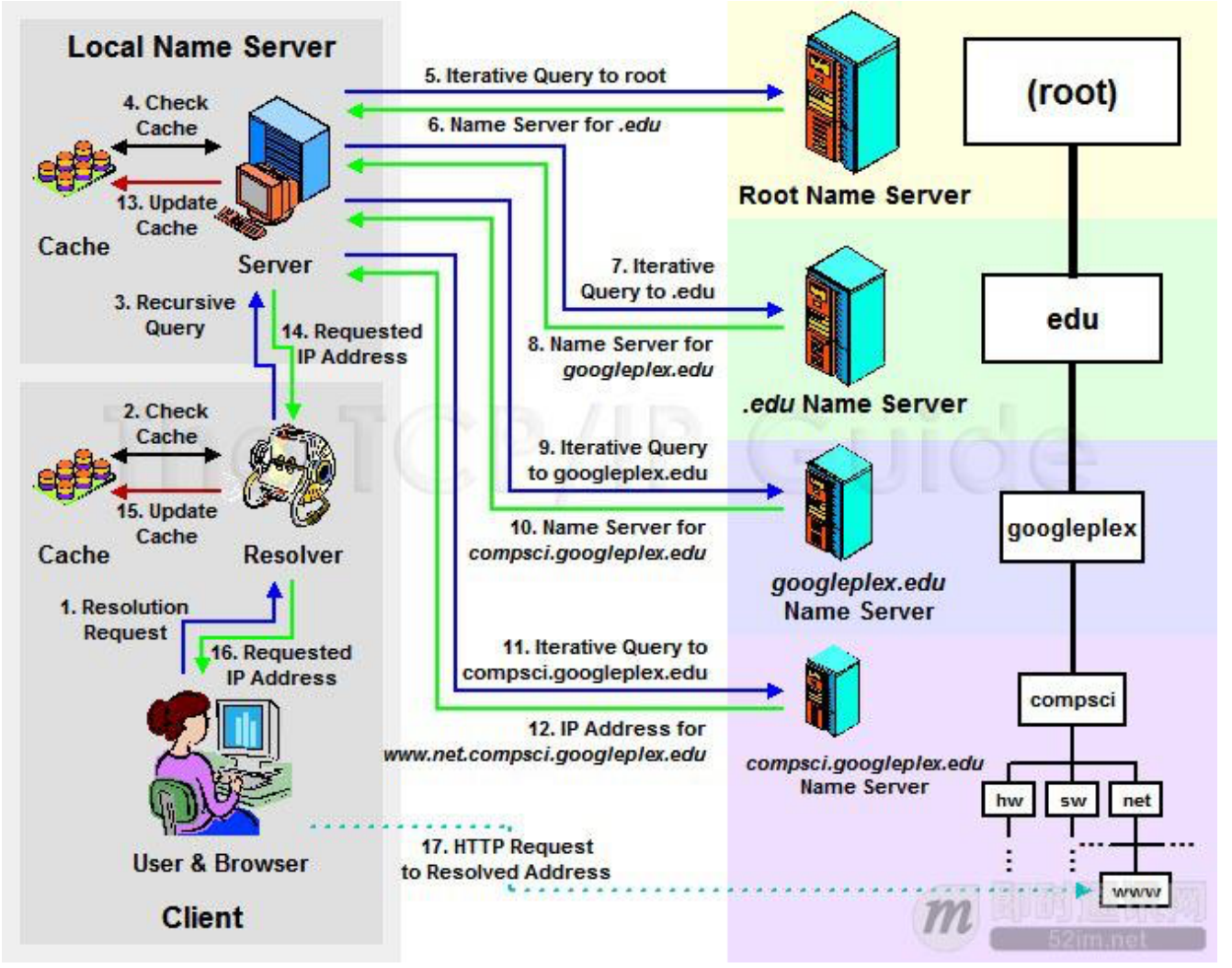
4.2DNS 域名系统结构



如上图所示，典型DNS域名系统的结构如下：

- 1）Root 域名：DNS 域名使用时，规定由尾部句号来指定名称位于根或更高级别的域层次结构；
- 2）Top Level 顶级域名：用来指示某个国家、地区或组织使用的名称的类型名称。如 .net；
- 3）Second Level 域名：个人或组织在 Internet 上使用的注册名称。如 52im.net；
- 4）Third Level 域名：已注册的二级域名派生的域名。如 docs.52im.net。

4.3DNS 解析过程



如上图所示，这是一个典型的域名解析过程：

- 1）浏览器中输入 www.52im.net，发出解析请求；
- 2）本机的域名解析器 resolver 程序查询本地缓存和 host 文件中是否为域名的映射关系，如果有则调用这个 IP 地址映射，完成解析；
- 3）如果 hosts 与本地解析器缓存都没有相应的网址映射关系，则本地解析器会向 TCP/IP 参数中设置的首选 DNS 服务器（我们叫它 Local DNS 服务器）发起一个递归的查询请求；
- 4）服务器收到查询时，如果要查询的域名由本机负

责解析，则返回解析结果给客户机，完成域名解析，此解析具有权威性。如果要查询的域名，不由 Local DNS 服务器解析，但该服务器已缓存了此网址映射关系，则调用这个 IP 地址映射，完成域名解析，此解析不具有权威性；

- 5）如果 Local DNS 服务器本地区域文件与缓存解析都失效，则根据 Local DNS 服务器的设置（是否递归）进行查询，如果未用开启模式，Local DNS 就把请求发至13台 Root DNS。如果用的是递归模式，此 DNS 服务器就会把请求转发至上一级 DNS 服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根 DNS 或把转请求转至上上级，以此循环；
- 6）Root DNS 服务器收到请求后会判断这个域名是谁来授权管理，并会返回一个负责该顶级域名服务器的一个 IP；
- 7）Local DNS 服务器收到 IP 信息后，将会联系负责 .net 域的这台服务器；
- 8）负责 .com 域的服务器收到请求后，如果自己无法解析，它就会找一个管理 .net 域的下一级 DNS 服务器地址给本地 DNS 服务器；
- 9）当 Local DNS 服务器收到这个地址后，就会找 52im.net 域服务器，10、11重复上面的动作，进行查询；
- 10）最后 www.52im.net 返回需要解析的域名的 IP 地址给 Local DNS 服务器；
- 11）Local DNS 服务器缓存这个解析结果（同时也会缓存，6、8、10返回的结果）；
- 12）Local DNS 服务器同时将结果返回给本机域名解

析器；

- 13）本机缓存解析结果；
- 14）本机解析器将结果返回给浏览器；
- 15）浏览器通过返回的 IP 地址发起请求。

4.4DNS的递归查询和迭代查询

递归查询：如果主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其他根域名服务器继续发出查询请求报文，而不是让该主机自己进行下一步的查询。

迭代查询：当根域名服务器收到本地域名服务器发出的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器：你下一步应当向哪一个域名服务器进行查询。然后让本地域名服务器进行后续的查询，而不是替本地域名服务器进行后续的查询。

由此可见，客户端到 Local DNS 服务器，Local DNS 与上级 DNS 服务器之间属于递归查询；DNS 服务器与根 DNS 服务器之前属于迭代查询。

实际环境中，因为采用递归模式会导致 DNS 服务器流量很大，所以现在大多数的 DNS 都是迭代模式。

5、国内移动端网络所面临的各种DNS杂症

总结下来，DNS的这些咋整主要的带来了三类问题：

- 1）LocalDNS劫持；
- 2）平均访问延迟下降；
- 3）用户连接失败率下降。

LocalDNS劫持： 由于HttpDNS是通过ip直接请求http获取服务器A记录地址，不存在向本地运营商询问domain解析过程，所以从根本避免了劫持问题。（对于http内容tcp/ip层劫持，可以使用验证因子或者数据加密等方式来保证传输数据的可信度）

平均访问延迟下降： 由于是ip直接访问省掉了一次domain解析过程，（即使系统有缓存速度也会稍快一些‘毫秒级’）通过智能算法排序后找到最快节点进行访问。

用户连接失败率下降： 通过算法降低以往失败率过高的服务器排序，通过时间近期访问过的数据提高服务器排序，通过历史访问成功记录提高服务器排序。如果ip(a)访问错误，在下一次返回ip(b)或者ip(c) 排序后的记录。

那么，追根溯源，到底为什么会存在这些问题？这就是下一节要讨论的问题。

6、追根溯源，国内DNS问题的根源是什么？

我们得先得了解下现在国内各ISP运营商的LocalDNS的基本情况。

国内运营商LocalDNS造成的这些问题，可以归为下以下3种原因：

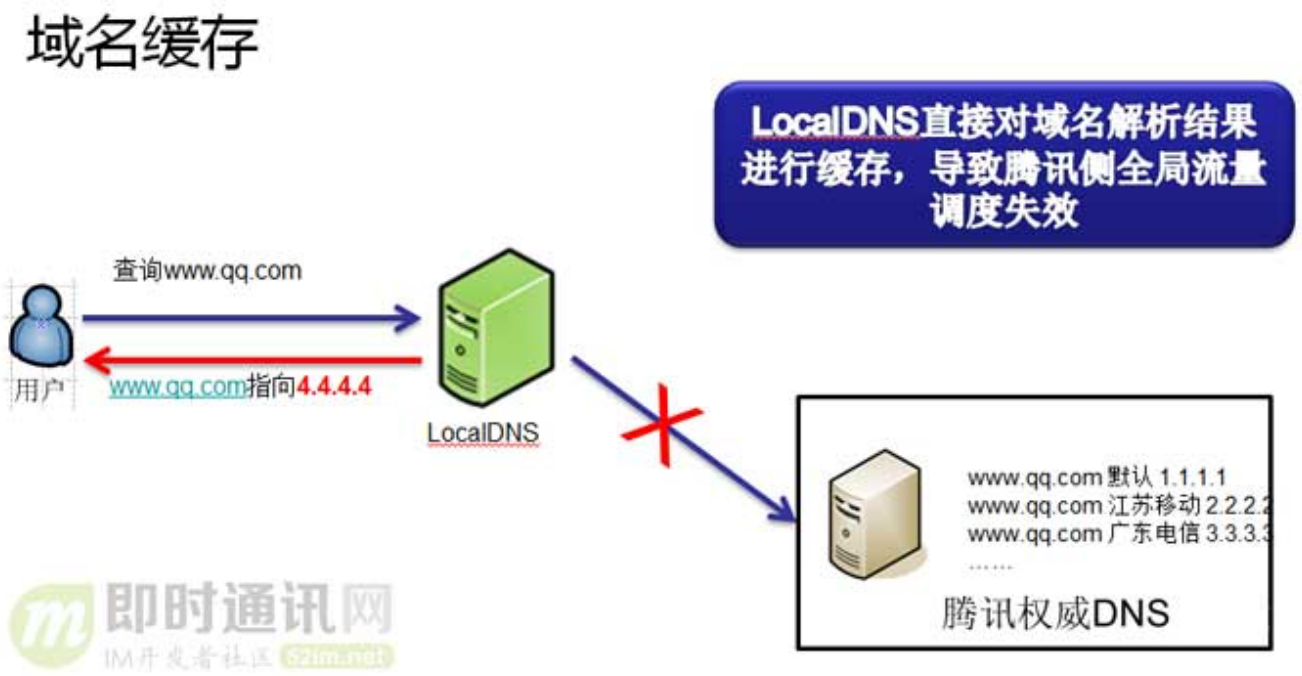
- 域名缓存；
- 解析转发；
- LocalDNS递归出口NAT。

下面，我们来逐一分析。

6.1域名缓存

域名缓存很好理解，就是LocalDNS缓存了腾讯的域名的解析结果，不向腾讯权威DNS发起递归。

示意图如下：



为何LocalDNS要把域名解析结果进行缓存呢？原因有以下几个：

- 1）保证用户访问流量在本网内消化：国内的各互联网接入运营商的带宽资源、网间结算费用、IDC机房分布、网内ICP资源分布等存在较大差异。为了保证网内用户的访问质量，同时减少跨网结算，运营商在网内搭建了内容缓存服务器，通过把域名强行指向内容缓存服务器的IP地址，就实现了把本地本网流量完全留在了本地的目的；
- 2）推送广告：有部分LocalDNS会把部分域名解析结果的所指向的内容缓存，并替换成第三方广告联盟的广告。

以上类型的行为就是我们常说的域名缓存，域名缓存会导致用户产生以下的访问异常：

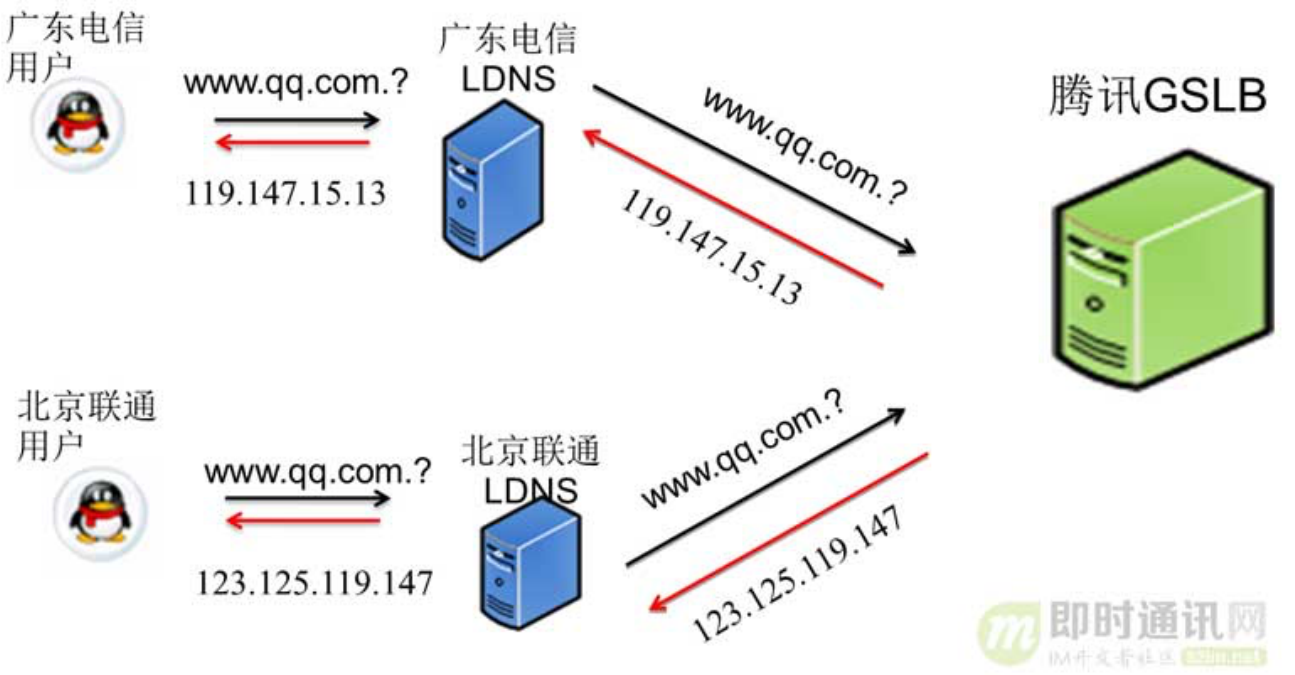
- A、仅对80端口的http服务做了缓存，如果域名是通过https协议或其它端口提供服务的，用户访问就会出现失败。比如支付服务、游戏通过指定端口连接connect server服务等；
- B、缓存服务器的运维水平参差不齐，时有出现缓存服务器故障导致用户访问异常的问题。

6.2解析转发

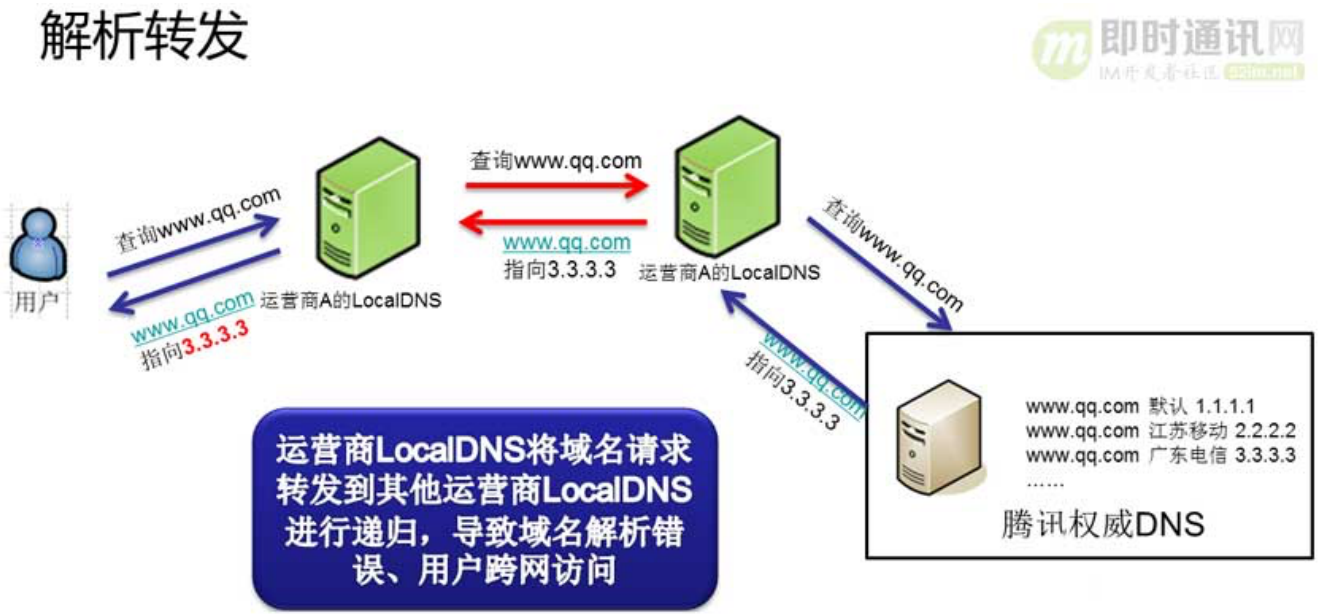
除了域名缓存以外，运营商的LocalDNS还存在解析转发的现象。解析转发是指运营商自身不进行域名递归解析，而是把域名解析请求转发到其它运营商的递归DNS上的行为。

正常的LocalDNS递归解析过程是这样的：

基于域名解析实现的GSLB



而部分小运营商为了节省资源，就直接将解析请求转发到了其它运营的递归LocalDNS上去了：



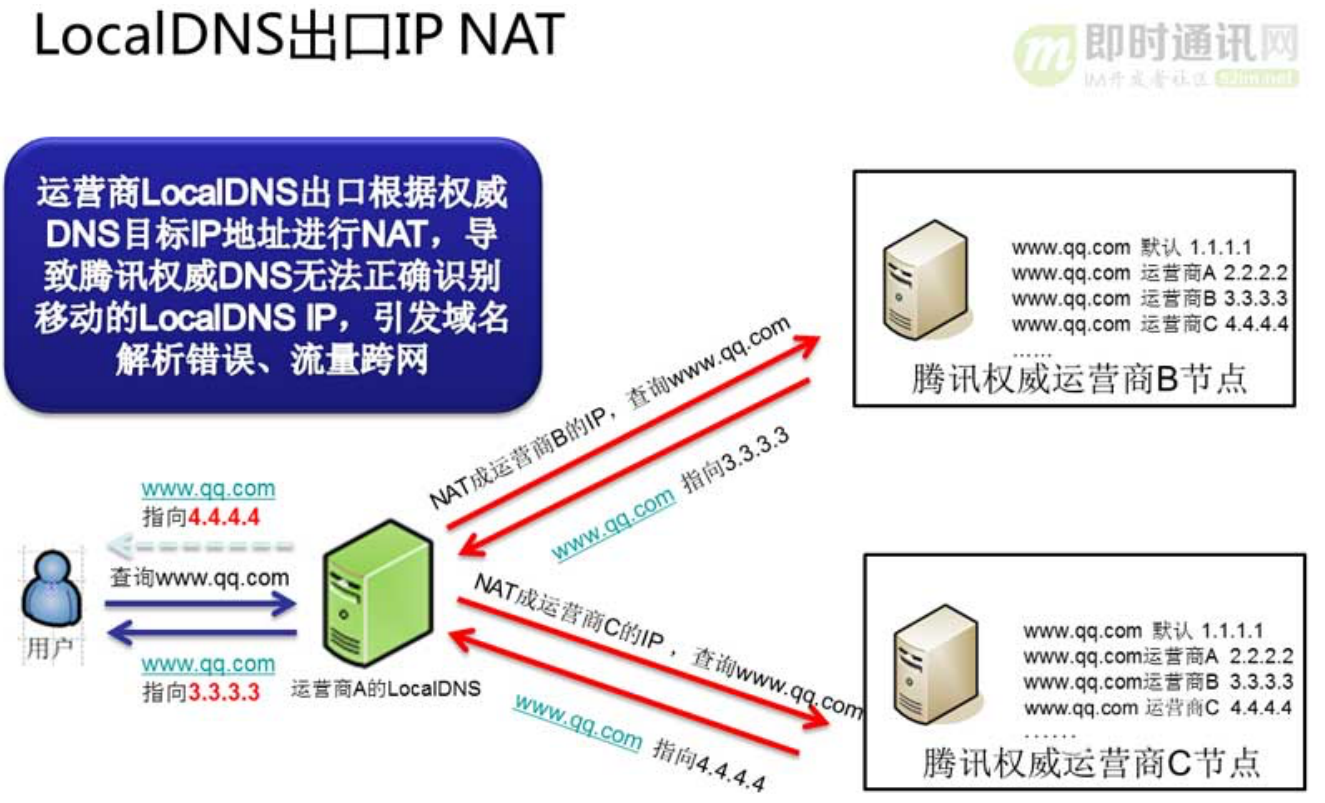
这样的直接后果就是腾讯权威DNS收到的域名解析请求的来源IP就成了其它运营商的IP，最终导致用户流量被导向

了错误的IDC，用户访问变慢。

6.3LocalDNS递归出口NAT

LocalDNS递归出口NAT指的是运营商的LocalDNS按照标准的DNS协议进行递归，但是因为在网络上存在多出口且配置了目标路由NAT，结果导致LocalDNS最终进行递归解析的时候的出口IP就有概率不为本网的IP地址。

如下图所示：



这样的直接后果就是GSLB DNS收到的域名解析请求的来源IP还是成了其它运营商的IP，最终导致用户流量被导向了错误的IDC，用户访问变慢。

7、必须着手解决这些问题，但传统解决方案问题太多

运营商的LocalDNS解析域名异常，给对用户访问腾讯互联网业务的体验造成了非常大的损害。

那么以前，我们是如何处理这些域名解析异常的问题的呢?

1）实时监控+商务推动：

这种方案是目前腾讯的运营团队一直在使用的方案。这种方案就是周期比较长，毕竟通过行政手段来推动运营商来解决这个问题是比较耗时的。另外我们通过大数据分析，得出的结论是Top 3的问题用户均为移动互联网用户。对于这部分用户，我们有什么技术手段可以解决以上的问题呢?

2）绕过自动分配DNS，使用114dns或Google public DNS：

这个方案看上去很美好，114dns是国内最大的中立缓存DNS，而Google又是秉承不作恶理念的互联网工程帝国巨鳄，而且腾讯的权威DNS又支持edns-client-subnet功能，能直接识别使用Google publicDNS解析腾讯域名的用户的IP地址，不会出现流量调度失效。

但是问题来了：

- a. 如何在用户侧构造域名请求：对于PC端的客户端来说，构造一个标准的DNS请求包并不算什么难事。但在移动端要向一个指定的LocalDNS上发送标准的DNS请求包，而且要兼容各种iOS和android的版本的话，技术上是可行的，只是兼容的成本会很高；
- b. 推动用户修改配置极高：如果要推动用户手动修改PC的DNS配置的话，在PC端和手机客户端的WiFI下面还算勉强可行。但是要用户修改在移动互联网环境下的DNS配置，其难度不言而喻。

3）完全抛弃域名，自建connectcenter进行流量调度：

如果要采用这种这种方案的话，首先你就得要拿到一份准确的IP地址库来判断用户的归属，然后再制定个协议搭个connect center来做调度，然后再对接入层做调度改造。这种方案和2种方案一样，不是不能做，只是成本会比较高，尤其对于腾讯这种业务规模如此庞大的公司而言。

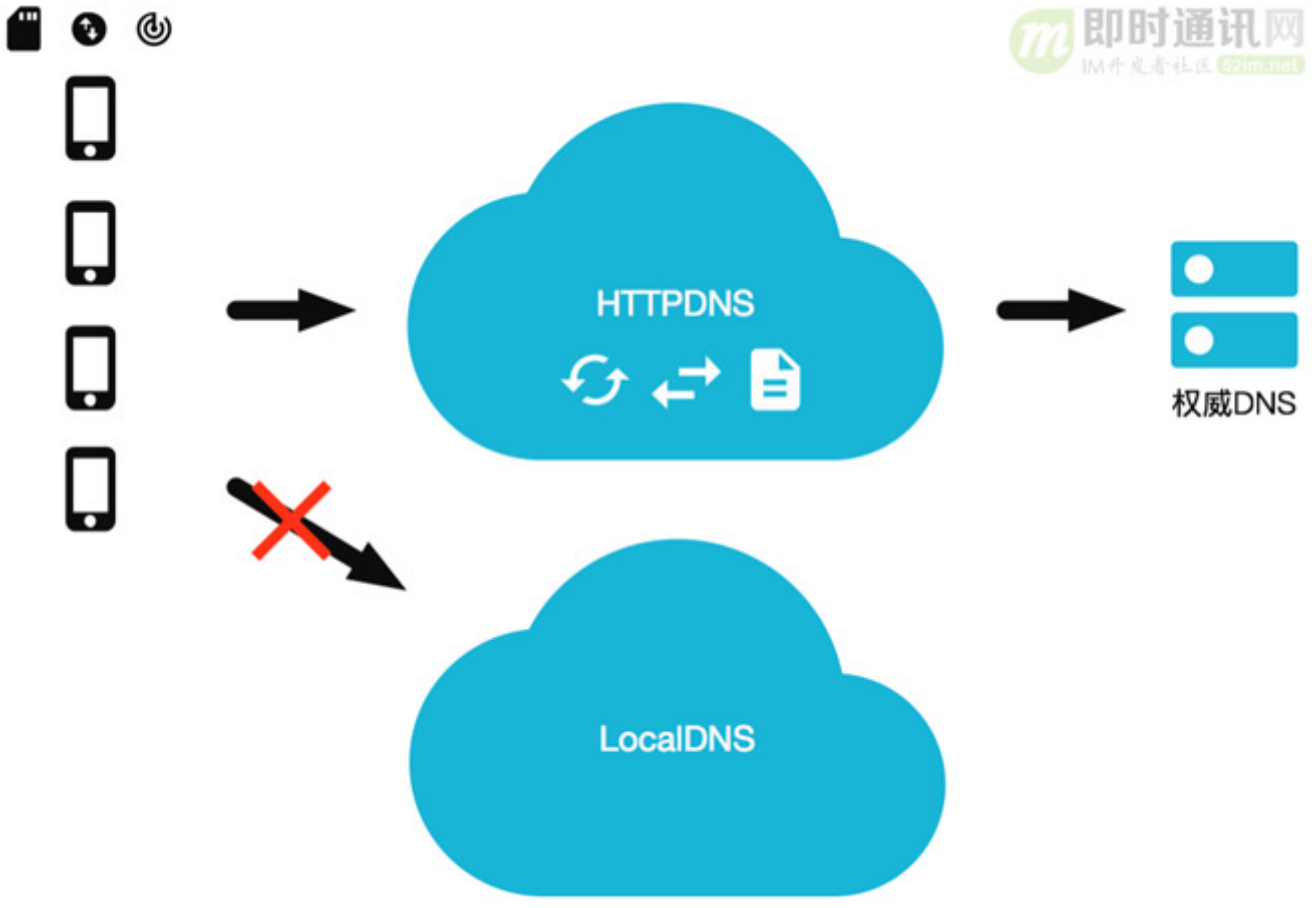
既然上面的这些传统方案都存在那么多的问题，那有没有一种调度精准、成本低廉、配置方便的基于域名的流量调度系统呢？答案是肯定的，我们在下一步继续分享。

8、当前主流的解决方案：HttpDNS出现了

8.1什么HttpDNS?

HTTPDNS 利用 HTTP 协议与 DNS 服务器交互，代替了传统的基于 UDP 协议的 DNS 交互，绕开了运营商的 Local DNS，有效防止了域名劫持，提高域名解析效率。另外，由于 DNS 服务器端获取的是真实客户端 IP 而非 Local DNS 的 IP，能够精确定位客户端地理位置、运营商信息，从而有效改进调度精确性。

HTTPDNS的原理如下图所示：



8.2HttpDns 主要解决的问题

Local DNS 劫持：由于 HttpDns 是通过 IP 直接请求 HTTP 获取服务器 A 记录地址，不存在向本地运营商询问 domain 解析过程，所以从根本避免了劫持问题。

平均访问延迟下降：由于是 IP 直接访问省掉了一次 domain 解析过程，通过智能算法排序后找到最快节点进行访问。

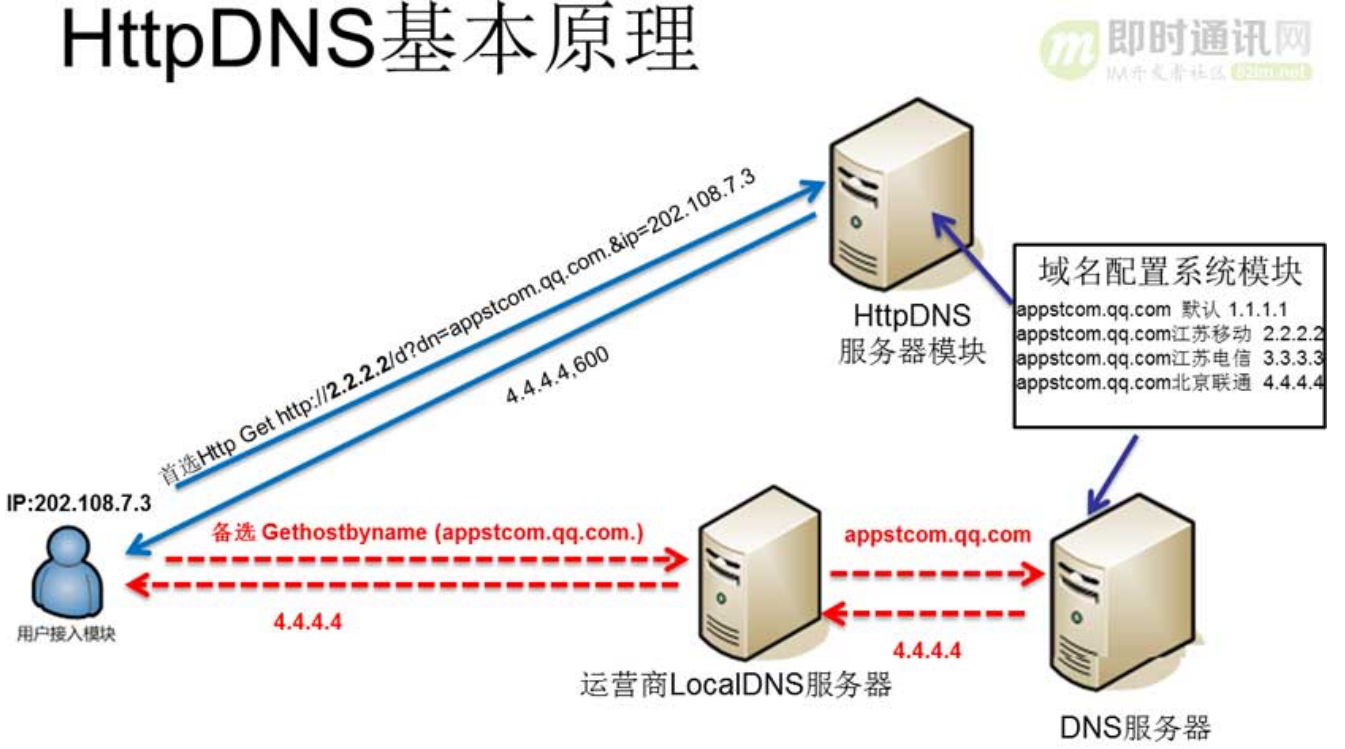
用户连接失败率下降：通过算法降低以往失败率过高的服务器排序，通过时间近期访问过的数据提高服务器排序，通过历史访问成功记录提高服务器排序。

8.3腾讯的HttpDNS思路

经过努力，腾讯公司的GSLB 团队推出的HttpDNS是为移动客户端量身定做的基于Http协议和域名解析的流量调度解决方案，专治LocalDNS解析异常以及流量调度不准。

详细介绍如下。

腾讯的HttpDNS基本原理：



HttpDNS的原理非常简单，主要有两步：

- A、客户端直接访问HttpDNS接口，获取业务在域名配置管理系统上配置的访问延迟最优的IP。（基于容灾考虑，还是保留次选使用运营商LocalDNS解析域名的方式）；
- B、客户端向获取到的IP后就向直接往此IP发送业务协议请求。以Http请求为例，通过在header中指定host字段，向HttpDNS返回的IP发送标准的Http请求即可。

HttpDNS带来的优势：

从原理上来讲，HttpDNS只是将域名解析的协议由DNS协议换成了Http协议，并不复杂。

但是这一微小的转换，却带来了无数的收益：

- A、根治域名解析异常：由于绕过了运营商的LocalDNS，用户解析域名的请求通过Http协议直接透传到了腾讯的HttpDNS服务器IP上，用户在客户端的域名解析请求将不会遭受到域名解析异常的困扰；
- B、调度精准：HttpDNS能直接获取到用户IP，通过结合腾讯自有专利技术生成的IP地址库以及测速系统，可以保证将用户引导的访问最快的IDC节点上；
- C、实现成本低廉：接入HttpDNS的业务仅需要对客户端接入层做少量改造，无需用户手机进行root或越狱；而且由于Http协议请求构造非常简单，兼容各版本的移动操作系统更不成问题；另外HttpDNS的后端配置完全复用现有权威DNS配置，管理成本也非常低。总而言之，就是以最小的改造成本，解决了业务遭受域名解析异常的问题，并满足业务精确流量调度的需求；
- D、扩展性强：HttpDNS提供可靠的域名解析服务，业务可将自有调度逻辑与HttpDNS返回结果结合，实现更精细化的流量调度。比如指定版本的客户端连接请求的IP地址，指定网络类型的用户连接指定的IP地址等。

当然各位可能会问：用户将首选的域名解析方式切换到了HttpDNS，那么HttpDNS的高可用又是如何保证的呢？另外不同运营商的用户访问到同一个HttpDNS的服务IP，用户的访问延迟如何保证？

为了保证高可用及提升用户体验，HttpDNS通过接入了腾讯公网交换平台的BGP Anycast网络，与全国多个主流运营商建立了BGP互联，保证了这些运营商的用户能够快速地访问到HttpDNS服务；另外HttpDNS在多个数据中心进行了部署，任意一个节点发生故障时均能无缝切换到备份节点，保证用户解析正常。

接入效果及未来展望：

当前腾讯的HttpDNS方案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。

以某个接入HttpDNS的业务为例，该业务仅通过接入HttpDNS，在未做任何其它优化的情况下，用户平均访问延迟下降超过10%，访问失败率下降了超过五分之一，用户访问体验的效果提升非常显著。另外腾讯的HttpDNS服务除了在腾讯内部被广泛使用以外，也受到了业务同行的肯定。国内最大的publicDNS服务商114dns在受到腾讯DNS的启发下，也推出了HttpDNS服务。

在未来的日子里，腾讯GSLB团队将会在腾讯内部进一步推广HttpDNS服务，并将在实际业务的需求下对HttpDNS服务进行升级，如提供更为通用、安全、简单的接入协议，进一步提升接入用户的网络访问体验等等。希望HttpDNS能为各位在解决域名解析异常及全局流量调度失效方面提供一个简单、可行的思路。

9、作为创业团队，如何改造APP并支持HttpDNS?

作为创业团队，人力、财力、技术力量可能都无暇顾及这一块，但是移动端APP却实实在在面临文中提到的各种DNS问题，我们该怎么办呢?

9.1使用第3方云服务商提供的HttpDNS接口

目前，国内有一部分厂商已经提供了这个解析服务，可以直接使用第三方服务。

目前，提供 HttpDns 解析服务的第3方服务商越来多，比如：阿里云HttpDNS、腾讯云HttpDNS、华为云HttpDNS等。

以阿里云的 HttpDNS为便，它的API 比较标准，直接发一个 Get 请求，带上请求参数，返回结果以 json 返回：

http://203.107.1.1/d?host=www.52im.net

请求成功时，返回结果如下：

```
{
  "host": "www.linkedkeeper.com",
  "ips": [
    "115.238.23.241",
```

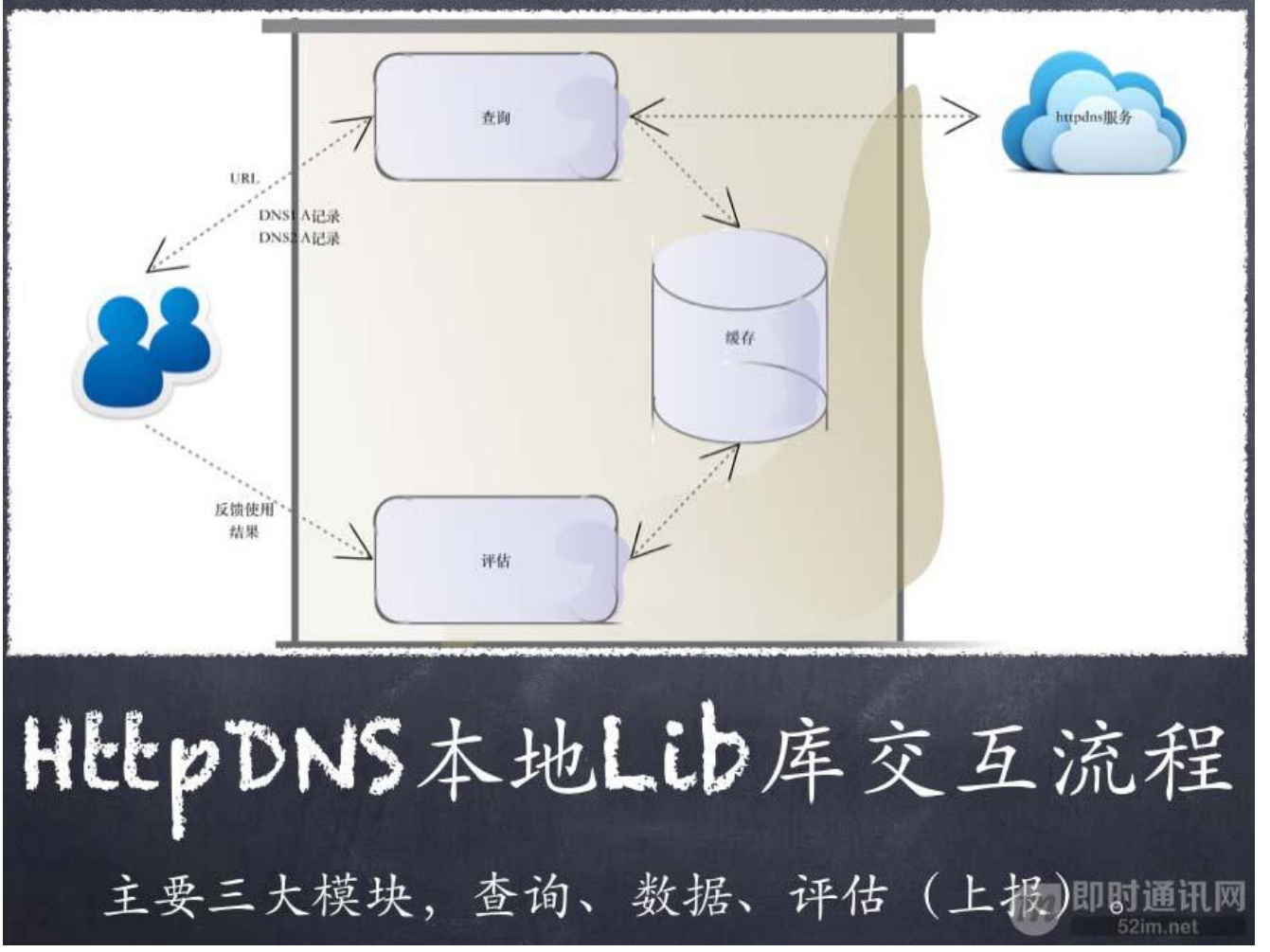
```
    "115.238.23.251"
  ],
  "ttl": 57
}
```

在移动端，将由 HttpDns 获得的 IP 地址在原有 URL 的基础上，将域名替换为 IP，然后用新的 URL 发起 HTTP 请求。

9.2新浪微博团队分享的开源HttpDNS库：HTTPDNSLib

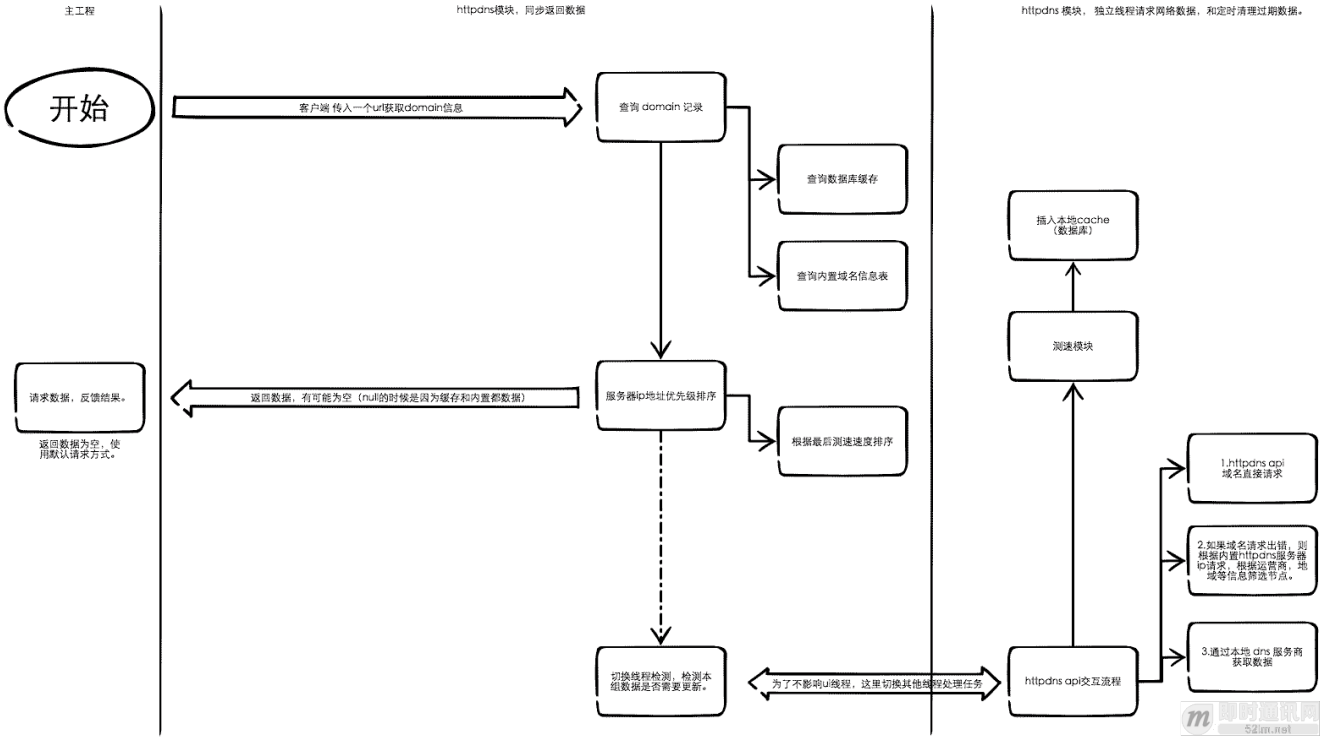
HTTPDNSLib的Github地址：https://github.com/CNSRE/HTTPDNSLib

HTTPDNSLib中实现的HttpDNS交互流程原理：



从上图中可以看出来 整个业务的交互流程，用户向查询模块传入一个URL地址，然后查询模块会检查缓存是否存在，不存在从httpdnsapi接口查询， 然后经过评估模块返回。在用户请求URL过程完毕时，需要将这次请求的结果反馈给 lib库的评估模块由评估模块入库记录本次质量数据。

HttpDns Lib库交互详细流程：



上图这张详细流程图就更深入的说了下 lib的工作原理。有两条竖线讲图片分为了三个区域，分别是左部分、中间部分 和 右部分。

左部分是app主线程操作的事情，中间部分是app调用者线程中处理lib库事件逻辑的事情，右面部分是新线程独立处理事件的逻辑。

开始是里客户端调用方，传入一个 url，获取domain信息后由查询模块查询domain记录，查询模块会从内存缓存层查询，内存缓存层没有数据会，查询数据库，如果数据库也没有数据会请求本地 LocalDNS。从三个环节中任何一个环节拿到数据后， 都会进入下一个环节，如果没有拿到数据返回null结束。进入评估模块，根据五个插件进行排序， 排序后返回数据给客户端。

lib模块设定定时器，根据ttl过期时间来检查domain是否需

要更新。 定时器是独立线程， 不会影响app主线程。 httpdns api请求数据， 先从自己配置的 httpdns api接口获取数据，如果获取不到会从 dnspod api接口获取如果也获取不到 直接从本地 localDNS获取数据，（从本地localDNS获取数据后期会改为发送UDP包封装dns协议从公共dns服务器直接获取，比如114dns等。dns服务器地址可自行设定。）获取到数据后进入测速模块。 测速模块最新版本可以配置两种方式，一种是http空请求。 两个http头的交互，类似tcp首保耗时时间原理 ，用来测试链路最快。 另一种是ping命令，（icmp协议）来尽量最小化流量的消耗，考虑倒可能有的服务器禁ping就使用空http测速即可。 测速后将数据插入本地 cache 即可。

下面是测试Demo的截图：

下午5:07

模拟任务

Local 推荐:123.126.157.222 √

模拟开始

2015-07-14 17:07:07

任务ID:8 延迟下降: 173

HTTP 推荐:111.161.68.60 √

Local 推荐:123.126.157.222 √

2015-07-14 17:07:07

任务ID:9 延迟下降: 71

HTTP 推荐:111.161.68.61 √

Local 推荐:123.126.157.222 √

2015-07-14 17:07:08

任务ID:10 延迟下降: -74

HTTP 推荐:111.161.68.61 √

Local 推荐:123.126.157.222 √

2015-07-14 17:07:08

模拟任务 数据配置 数据报表 缓存数据

下午5:11

详情页

任务

ID: 1

状态: 成功

开始时间: 2015-07-14 17:07:07

结束时间: 2015-07-14 17:07:07

消耗时间: 325毫秒

HTTPDNS 库

返回结果: 111.161.68.61、111.161.68.60、

响应时间: 7毫秒

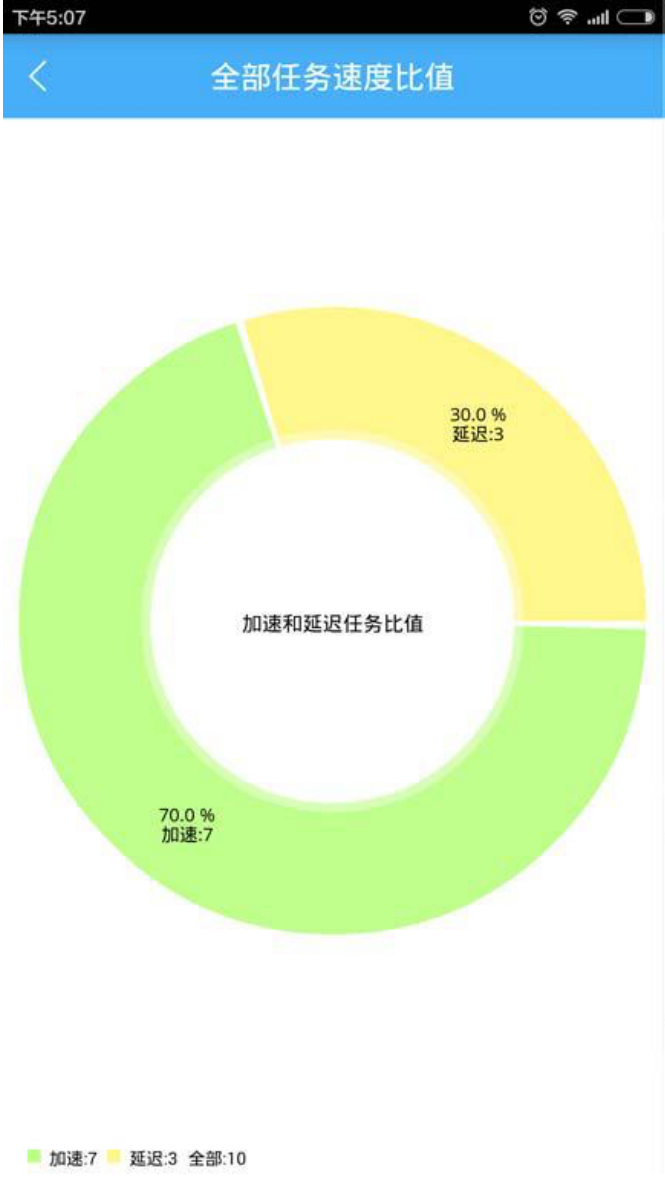
IP+HOST 请求

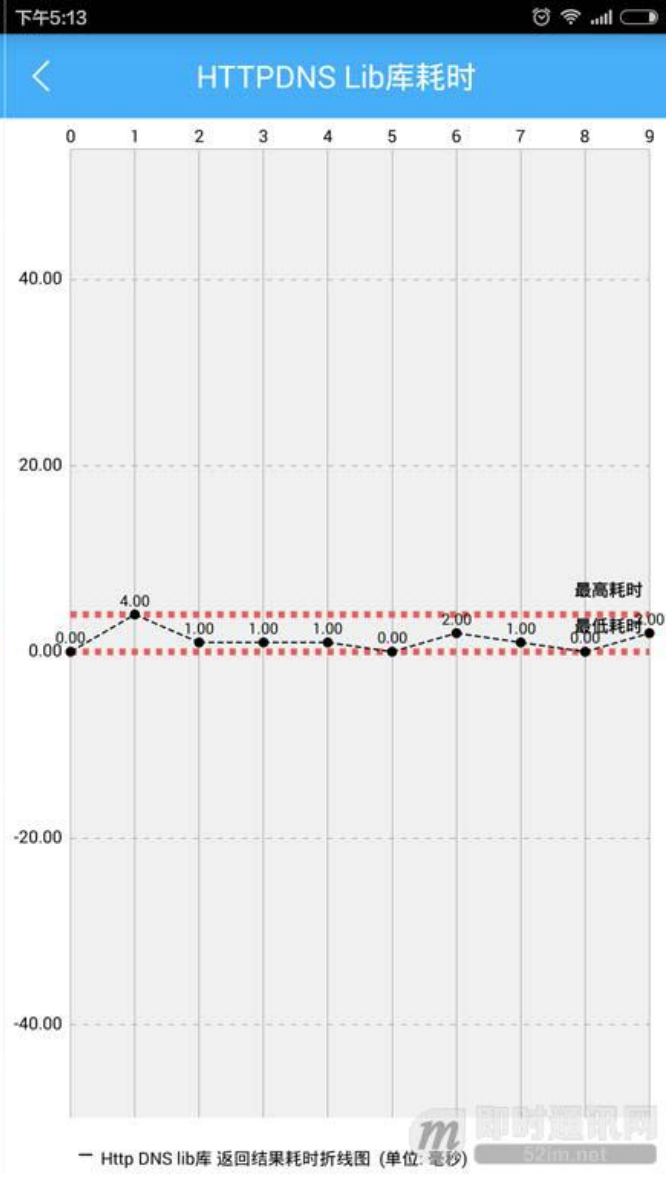
访问URL: http://111.161.68.61/webp360/40411806jw1et6q2zn9c6j20qo0zk4qp.jpg

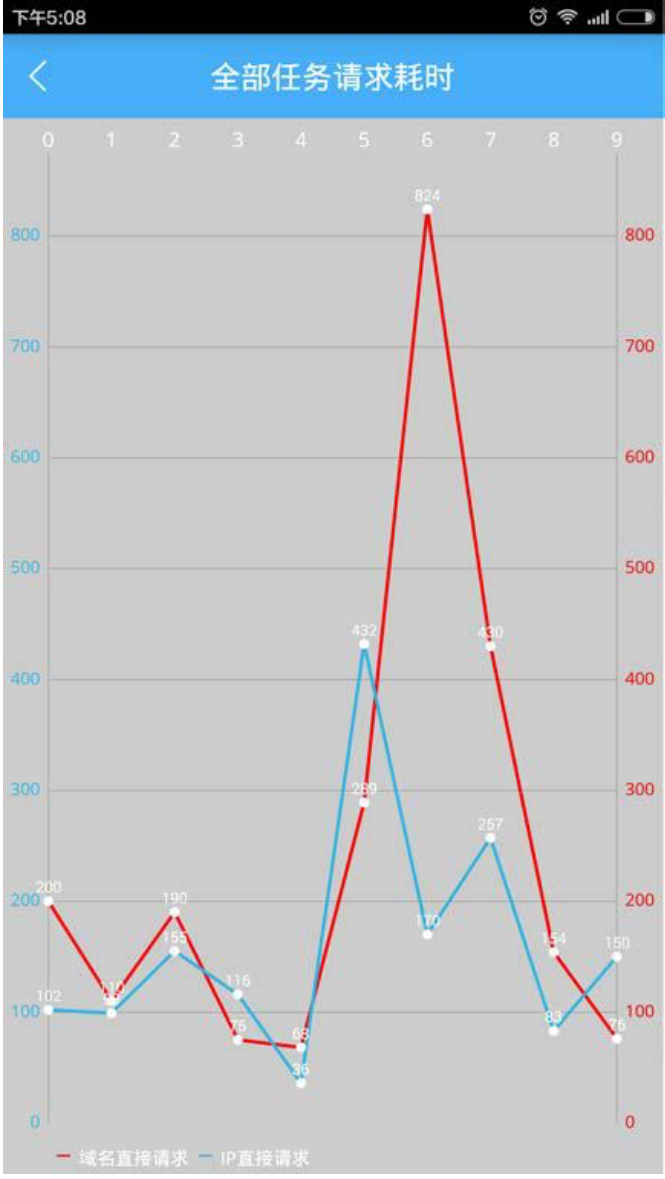
访问服务器IP: 111.161.68.61 [北京市 联通]

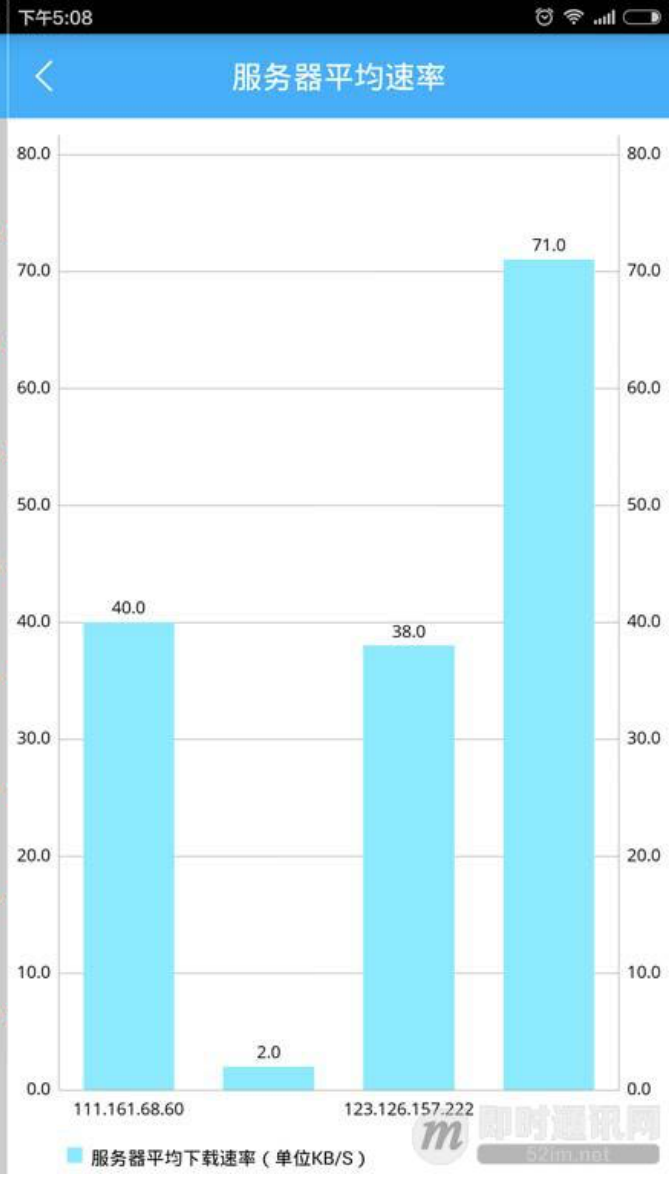
服务器返回CODE: 200

服务器返回结果:


下午5:07
全部任务速度比值
30.0 %
延迟:3
加速和延迟任务比值
70.0 %
加速:7
加速:7 延迟:3 全部:10


下午5:13
HTTPDNS Lib库耗时
0 1 2 3 4 5 6 7 8 9
40.00
20.00
0.00
-20.00
-40.00
最高耗时
最低耗时
4.00
1.00
1.00
1.00
0.00
2.00
1.00
0.00
0.00
Http DNS lib库 返回结果耗时折线图 (单位: 毫秒)


下午5:08
全部任务请求耗时
域名直接请求
IP直接请求


下午5:08
服务器平均速率
111.161.68.60
123.126.157.222
服务器平均下载速率（单位KB/S）

有关HttpDns Lib库的详细介绍，请见：《App域名劫持之DNS高可用 - 开源版HttpDNS方案详解》。

附录：更多网络通信方面的精华文章

《TCP/IP详解 - 第11章·UDP：用户数据报协议》
《TCP/IP详解 - 第17章·TCP：传输控制协议》
《TCP/IP详解 - 第18章·TCP连接的建立与终止》
《TCP/IP详解 - 第21章·TCP的超时与重传》
《技术往事：改变世界的TCP/IP协议（珍贵多图、手机慎

点）》

《通俗易懂-深入理解TCP协议（上）：理论基础》

《通俗易懂-深入理解TCP协议（下）：RTT、滑动窗口、拥塞处理》

《理论经典：TCP协议的3次握手与4次挥手过程详解》

《理论联系实际：Wireshark抓包分析TCP 3次握手、4次挥手过程》

《计算机网络通讯协议关系图（中文珍藏版）》

《UDP中一个包的大小最大能多大？》

《P2P技术详解(一)：NAT详解——详细原理、P2P简介》

《P2P技术详解(二)：P2P中的NAT穿越(打洞)方案详解》

《P2P技术详解(三)：P2P技术之STUN、TURN、ICE详解》

《通俗易懂：快速理解P2P技术中的NAT穿透原理》

《高性能网络编程(一)：单台服务器并发TCP连接数到底可以有多少》

《高性能网络编程(二)：上一个10年，著名的C10K并发连接问题》

《高性能网络编程(三)：下一个10年，是时候考虑C10M并发问题了》

《高性能网络编程(四)：从C10K到C10M高性能网络应用的理论探索》

《高性能网络编程(五)：一文读懂高性能网络编程中的I/O模型》

《高性能网络编程(六)：一文读懂高性能网络编程中的线程模型》

《不为人知的网络编程(一)：浅析TCP协议中的疑难杂症(上篇)》

《不为人知的网络编程(二)：浅析TCP协议中的疑难杂症

(下篇)》
《不为人知的网络编程(三)：关闭TCP连接时为什么会TIME_WAIT、CLOSE_WAIT》
《不为人知的网络编程(四)：深入研究分析TCP的异常关闭》
《不为人知的网络编程(五)：UDP的连接性和负载均衡》
《不为人知的网络编程(六)：深入地理解UDP协议并用好它》
《不为人知的网络编程(七)：如何让不可靠的UDP变的可靠？》
《网络编程懒人入门(一)：快速理解网络通信协议（上篇）》
《网络编程懒人入门(二)：快速理解网络通信协议（下篇）》
《网络编程懒人入门(三)：快速理解TCP协议一篇就够》
《网络编程懒人入门(四)：快速理解TCP和UDP的差异》
《网络编程懒人入门(五)：快速理解为什么说UDP有时比TCP更有优势》
《网络编程懒人入门(六)：史上最通俗的集线器、交换机、路由器功能原理入门》
《网络编程懒人入门(七)：深入浅出，全面理解HTTP协议》
《网络编程懒人入门(八)：手把手教你写基于TCP的Socket长连接》
《网络编程懒人入门(九)：通俗讲解，有了IP地址，为何还要用MAC地址？》
《技术扫盲：新一代基于UDP的低延时网络传输层协议——QUIC详解》
《让互联网更快：新一代QUIC协议在腾讯的技术实践分

享》

《现代移动端网络短连接的优化手段总结：请求速度、弱网适应、安全保障》

《聊聊iOS中网络编程长连接的那些事》

《移动端IM开发者必读(一)：通俗易懂，理解移动网络的“弱”和“慢”》

《移动端IM开发者必读(二)：史上最全移动弱网络优化方法总结》

《IPv6技术详解：基本概念、应用现状、技术实践（上篇）》

《IPv6技术详解：基本概念、应用现状、技术实践（下篇）》

《从HTTP/0.9到HTTP/2：一文读懂HTTP协议的历史演变和设计思路》

《脑残式网络编程入门(一)：跟着动画来学TCP三次握手和四次挥手》

《脑残式网络编程入门(二)：我们在读写Socket时，究竟在读写什么？》

《脑残式网络编程入门(三)：HTTP协议必知必会的一些知识》

《脑残式网络编程入门(四)：快速理解HTTP/2的服务器推送(Server Push)》

《脑残式网络编程入门(五)：每天都在用的Ping命令，它到底是什么？》

《脑残式网络编程入门(六)：什么是公网IP和内网IP？NAT转换又是什么鬼？》

《以网游服务端的网络接入层设计为例，理解实时通信的技术挑战》

《迈向高阶：优秀Android程序员必知必会的网络基础》

《全面了解移动端DNS域名劫持等杂症：技术原理、问题根源、解决方案等》

《美图App的移动端DNS优化实践：HTTPS请求耗时减小近半》

>> 更多同类文章 ……