百度App网络深度优化系列《一》DNS优化

一、前言

网络优化是客户端几大技术方向中公认的一个深度领域，所以百度App给大家带来网络深度优化系列文章，其中包含系列《一》DNS优化，系列《二》连接优化，系列《三》弱网优化，希望对大家在网络方向的学习和实践有所帮助。

百度起家于搜索，整个公司的网络架构和部署都是基于标准的internet协议，目前已经是全栈HTTPS，来到移动互联网时代后，总的基础架构不变，但在客户端上需要做很多优化工作。

DNS（Domain Name System），它的作用是根据域名查出IP地址，它是HTTP协议的前提，只有将域名正确的解析成IP地址后，后面的HTTP流程才能进行，所以一般做网络优化会首选优化DNS。

二、背景

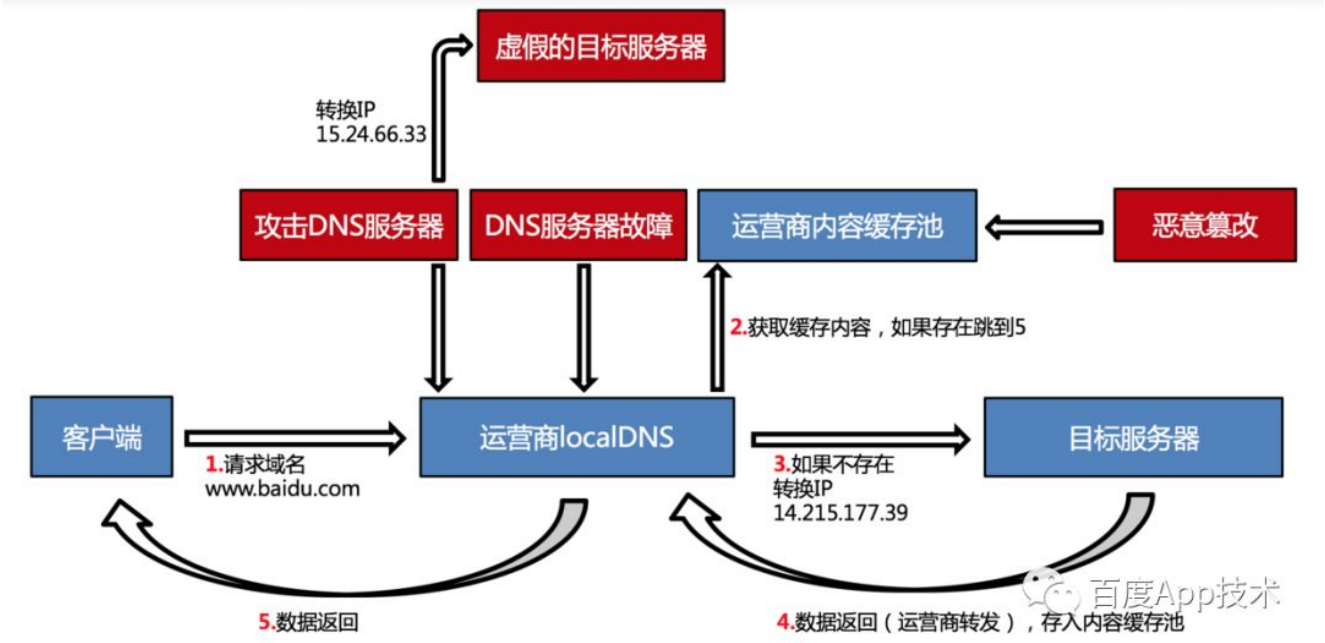
DNS优化核心需要解决的问题有两点：

【1】由于DNS劫持或故障造成的服务不可用，进而影响用户体验，影响公司的收入。

【2】由于DNS调度不准确导致的性能退化，进而影响用

户体验。

百度App承载着亿级流量，每年都会遇到运营商DNS劫持或运营商DNS故障，整体影响非常不好，所以DNS优化刻不容缓，通过下图会更直观的了解运营商劫持或故障的原理。

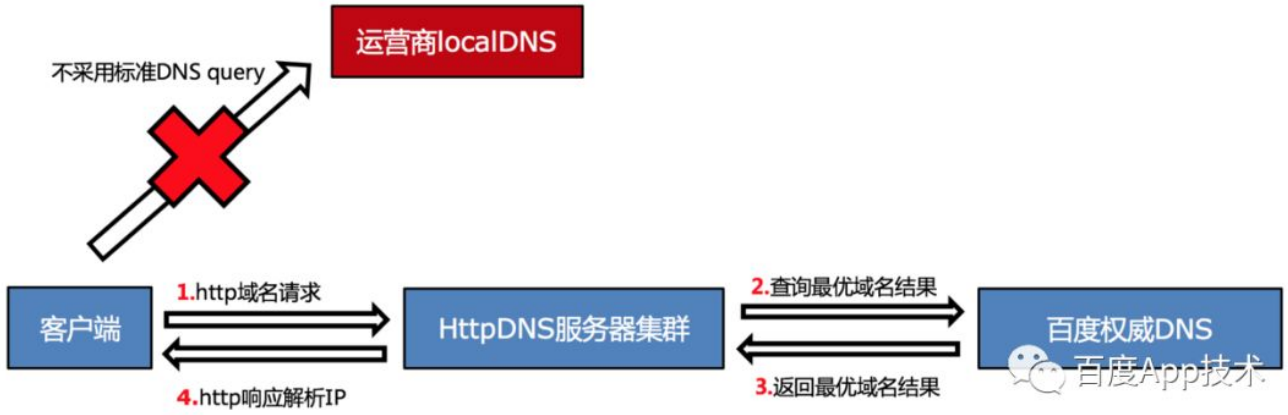


运营商劫持或故障的原理

三、HTTPDNS

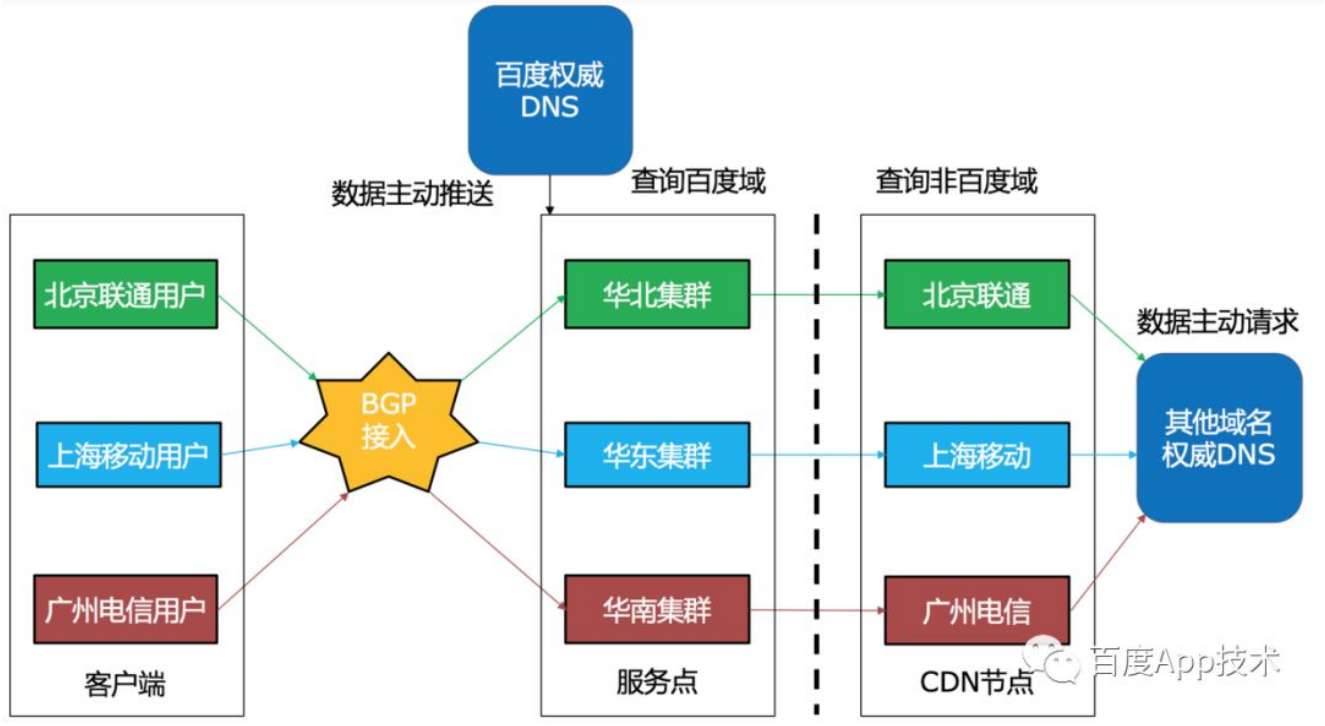
既然我们面临这么严峻的问题，那么我们如何优化DNS呢？答案就是HTTPDNS。

大部分标准DNS都是基于UDP与DNS服务器交互的，HTTPDNS则是利用HTTP协议与DNS服务器交互，绕开了运营商的Local DNS服务，有效防止了域名劫持，提高域名解析效率，下图是HTTPDNS的原理。



HTTPDNS原理

百度App HTTPDNS端上的实现是基于百度SYS团队的HTTPDNS服务，下图介绍了HTTPDNS的服务端部署结构。

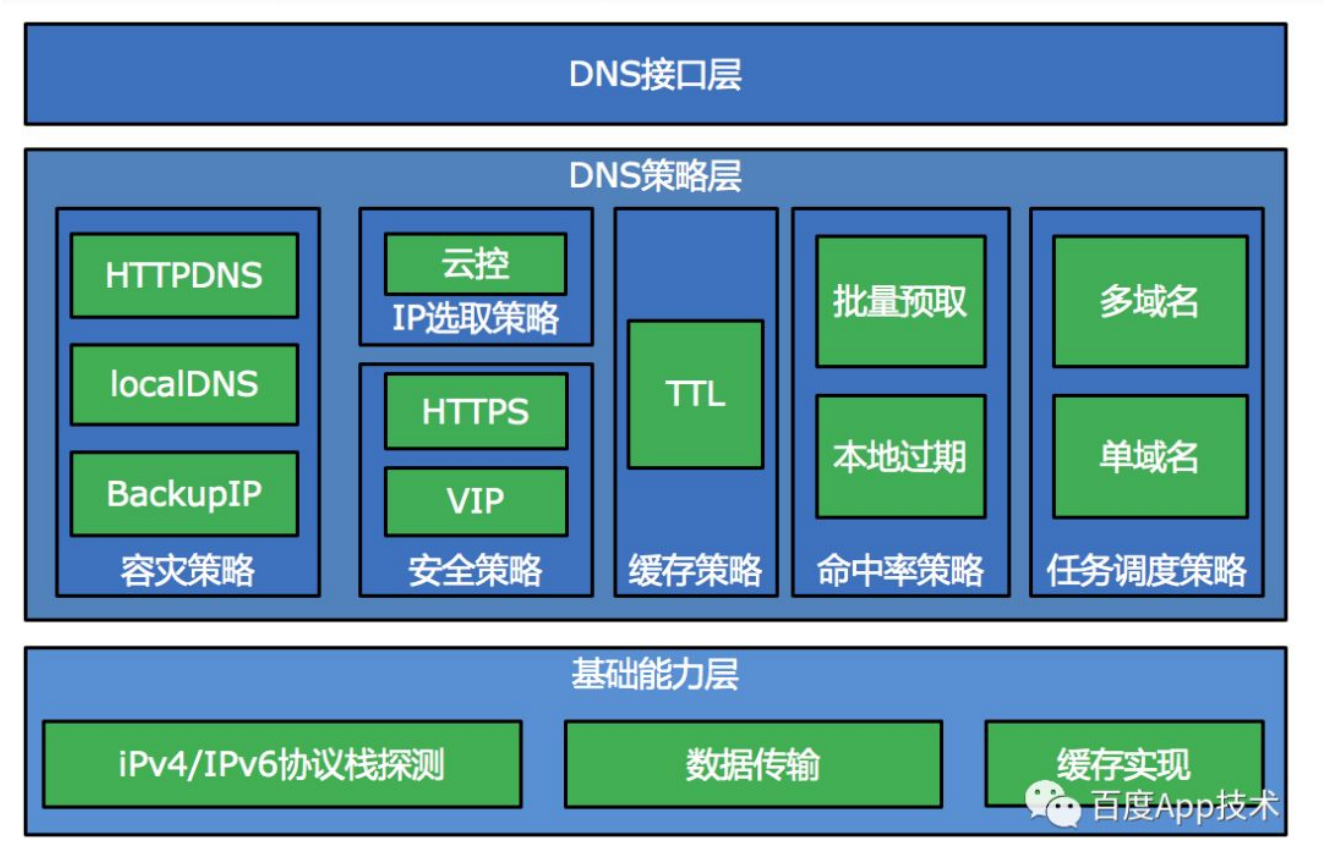


HTTPDNS部署结构

HTTPDNS服务是基于BGP接入的，BGP英文Border

Gateway Protocol，即边界网关协议，是一种在自治系统之间动态的交换路由信息的路由协议，BGP可以根据当前用户的运营商路由到百度服务点的对应集群上，对于第三方域名，服务点会通过百度部署在运营商的CDN节点向其他域名权威DNS发起查询，查询这个运营商下域名的最优IP。

百度App独立实现了端的HTTPDNS SDK，下图介绍了端HTTPDNS的整体架构。



端HTTPDNS的整体架构

DNS接口层：

DNS接口层解决的问题是屏蔽底层的细节，对外提供简单整洁的API，降低使用者的上手成本，提高开发效率。

DNS策略层：

DNS策略层通过多种策略的组合，使HTTPDNS服务在性能，稳定性，可用性上均保持较高的水准，下面讲解下每个策略设计的初衷和具体实现。

1.容灾策略

这是一个非常关键的策略，主要解决HTTPDNS服务可用性的问题，实践证明，这个策略帮助百度App在异常情况下挽救回很多流量。

【1】当HTTPDNS服务不可用并且本地也没有缓存或者缓存失效的时候，会触发降级策略，降级成运营商的localDNS方案，虽然存在运营商事故或者劫持的风险，但保障了DNS服务的可用性。

【2】当HTTPDNS服务和localDNS服务双双不可用的情况下，会触发backup策略，使用端上的backup IP。

什么是backup IP？backup IP是多组根据域名分类的IP列表，可云端动态更新，方便后续运维同学调整服务端的节点IP，不是所有域名都有对应的backup IP列表，目前百度App只能保证核心域名的可用性。

既然是一组IP，便有选取问题，backup IP选取机制是怎样的呢？我们的中心思想就是要在端上利用最小的代价，并且考虑服务端的负载均衡，得到相对正确或者合理的选取结果。通过运营商和地理信息，可以选择一个相对较优的IP，但获取地理信息需要很大耗时，外加频次很高，代价很大，所以我们选择了RR算法来代替上面的方法（RR算

法是Round-Robin，轮询调度），这样客户端的代价降低到最小，服务端也实现了负载均衡。

2.安全策略

【1】HTTPDNS解决的核心问题就是安全，标准的DNS查询大部分是基于UDP的，但也有基于TCP的，如果UDP被封禁，就需要使用TCP。不管是UDP还是TCP，安全性都是没有保障的，HTTPDNS查询是基于标准的HTTP协议，为了保证安全我们会在HTTP上加一层TLS（安全传输层协议），这便是HTTPS。

【2】解决了传输层协议的安全性后，我们要解决下域名解析的问题，上面我们提到HTTPDNS服务是基于BGP接入的，在端上采用VIP方式请求HTTPDNS数据（VIP即Virtual IP，VIP并没有与某设备存在必定的绑定关系，会跟随主备切换之类的情况发生而变换，VIP提供的服务是对应到某一台或若干台服务器的），既然请求原始数据需要使用IP直连的方式，那么就摆脱了运营商localDNS的解析限制，这样即使运营商出现了故障或者被劫持，都不会影响百度App的可用性。

3.任务调度策略

HTTPDNS服务提供了两类HTTP接口，用于请求最优域名结果。第一种是多域名接口，针对不同的产品线，下发产品线配置的域名，第二种是单域名接口，只返回你要查询的那个域名结果，这样的设计和标准的DNS查询基本是一样的，只不过是从UDP协议变成了HTTP协议。

【1】多域名接口会在App冷启动和网络切换的时候请求一次，目的是在App的网络环境初始化或者变化的时候预先获取域名结果，这样也会减少单域名接口的请求次数。

【2】单域名接口会在本地cache过期后，由用户的操作触发网络请求，进而做一次单域名请求，用户这次操作的DNS结果会降级成localDNS的结果，但在没有过期的情况下，下次会返回HTTPDNS的结果。

4.IP选取策略

IP选取策略解决的核心问题是最优IP的选取，避免因为接入点的选取错误造成的跨运营商耗时。HTTPDNS服务会将最优IP按照顺序下发，客户端默认选取第一个，这里没有做客户端的连通性校验的原因，主要还是担心端上的性能问题，不过有容灾策略兜底，综合评估还是可以接受的。

5.缓存策略

大家对于DNS缓存并不陌生，它主要是为了提升访问效率，操作系统，网络库等都会做DNS缓存。

DNS缓存中一个重要的概念就是TTL（Time-To-Live），在localDNS中针对不同的域名，TTL的时间是不一样的，在HTTPDNS中这个值由服务端动态下发，百度App目前所有的域名TTL的配置是5分钟，过期后如果没有新的IP将继续沿用老的IP，当然也可以选择不沿用老的IP，而降级成localDNS的IP，那么这就取决于localDNS对于过期IP的处理。

6.命中率策略

如果HTTPDNS的命中率是100%，在保证HTTPDNS服务稳定高效的前提下，我们就可以做到防劫持，提升精准调度的能力。

【1】为了提升HTTPDNS的命中率，我们选择使用多域名接口，在冷启动和网络切换的时候，批量拉取域名结果并缓存在本地，便于接下来的请求使用。

【2】为了再一次提升HTTPDNS的命中率，当用户操作触发网络请求，获取域名对应的IP时，会提前进行本地过期时间判断，时间是60s，如果过期，会发起单域名的请求并缓存起来，这样会持续延长域名结果的过期时间。本地过期时间与上面提到的TTL是客户端和服务端的双重过期时间，目的是在异常情况下可以双重保证过期时间的准确性。

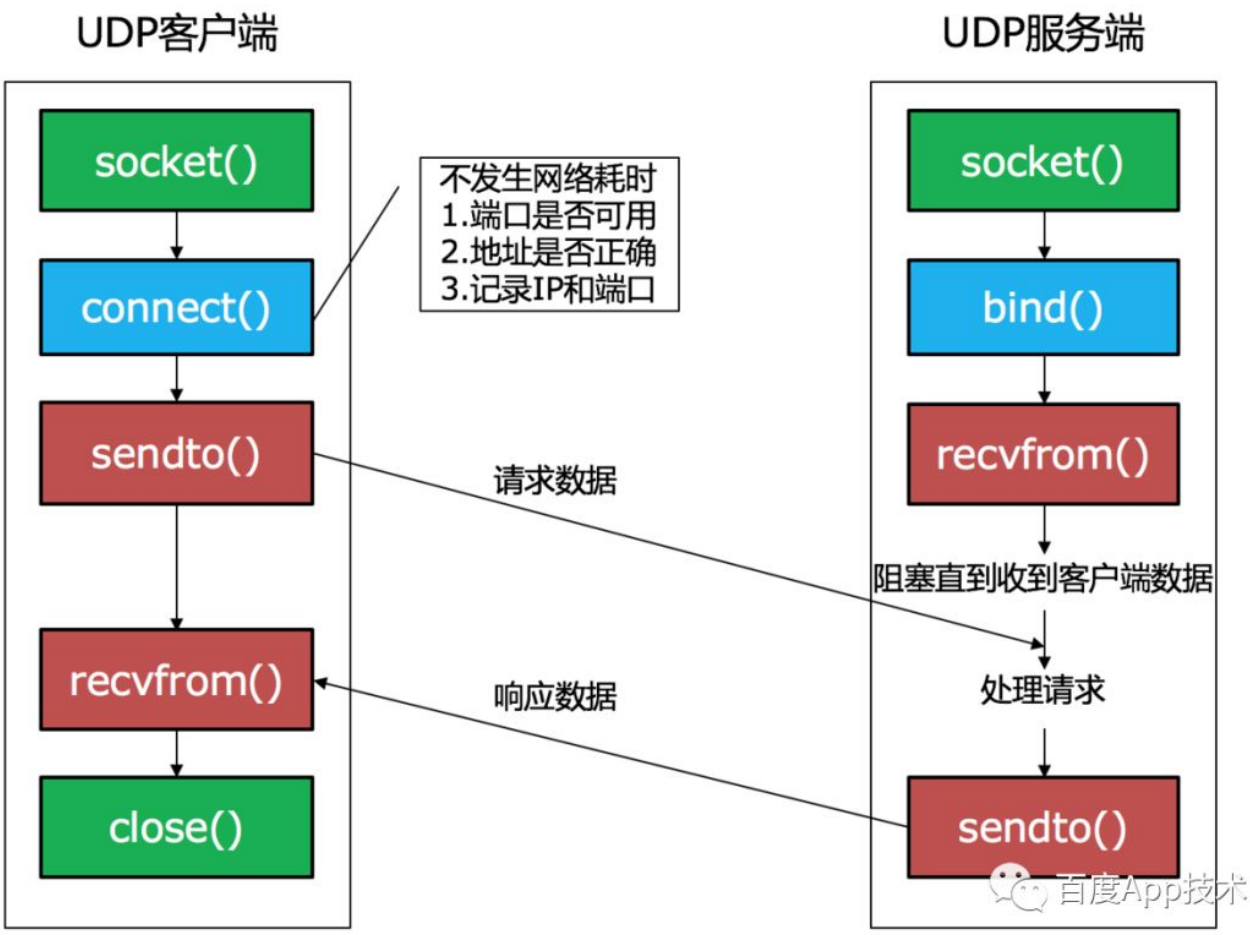
基础能力层：

基础能力层主要提供给DNS策略层所需要的基础能力，包括IPv4/IPv6协议栈探测的能力，数据传输的能力，缓存实现的能力，下面将讲解每种能力的具体实现

1.IPv4/IPv6协议栈探测：

百度App的IPv6改造正在如火如荼的进行中，端上在HTTPDNS的IP选取上如何知道目前属于哪个协议栈成为关键性问题，并且这种判断要求性能极高，因为IP选取的频次实在是太高了。

我们选取的方案是UDP Connect，那么何为UDP Connect？大家都知道TCP是面向连接的，传输数据前客户端都要调用connect方法通过三次握手建立连接，UDP是面向无连接的，无需建立连接便能收发数据，但是如果我们调用了UDP的connect方法会发生什么呢？当我们调用UDP的connect方法时，系统会检测其端口是否可用，地址是否正确，然后记录对端的IP地址和端口号，返回给调用者，所以UDP Connect不会像TCP Connect发起三次握手，发生网络真实损耗，UDP客户端只有调用send或者sendto方法后才会真正发起真实网络损耗。



UDP Connect原理

有了UDP Connect的基础保障，我们在上层做了缓存机

制，用来减少系统调用的损耗，时机上目前仅在冷启动和网络切换会触发探测，在同一种网络制式下探测一次基本可以确保当前网络是IPv4栈还是IPv6栈。

目前百度App客户端对于IPv4/IPv6双栈的策略是保守的，仅在IPv6-only的情况下使用v6的IP，其余使用的都是v4的IP，双栈下的方案后续需要优化，业内目前标准的做法是happy eyeball算法，什么叫happy eyeball呢？就是不会因为IPv4或IPv6的故障问题，导致用户的眼球一直在等待加载或者出错，这就是happy eyeball名字的由来。happy eyeball有v1版本RFC6555和v2版本RFC8305，前者是Cisco提出来的，后者是苹果提出来的。happy eyeball解决的核心问题是，复杂环境下v4和v6 IP选取的问题，它是一套整体解决方案，对于域名查询的处理，地址的排序，连接的尝试等方面均做出了规定，感兴趣的同学可以查看参考资料里的【5】和【6】。

2.数据传输：

数据传输主要提供网络请求的能力和数据解析的能力。

【1】网络请求失败重试的机制，获取HTTPDNS结果的成功率会大大影响HTTPDNS的命中率，所以客户端会有一个三次重试的机制，保障成功率。

【2】数据解析异常的机制，如果获取的HTTPDNS的结果存在异常，将不会覆盖端上的缓存。

3.缓存实现：

缓存的实现基本可以分为磁盘缓存和内存缓存，对于

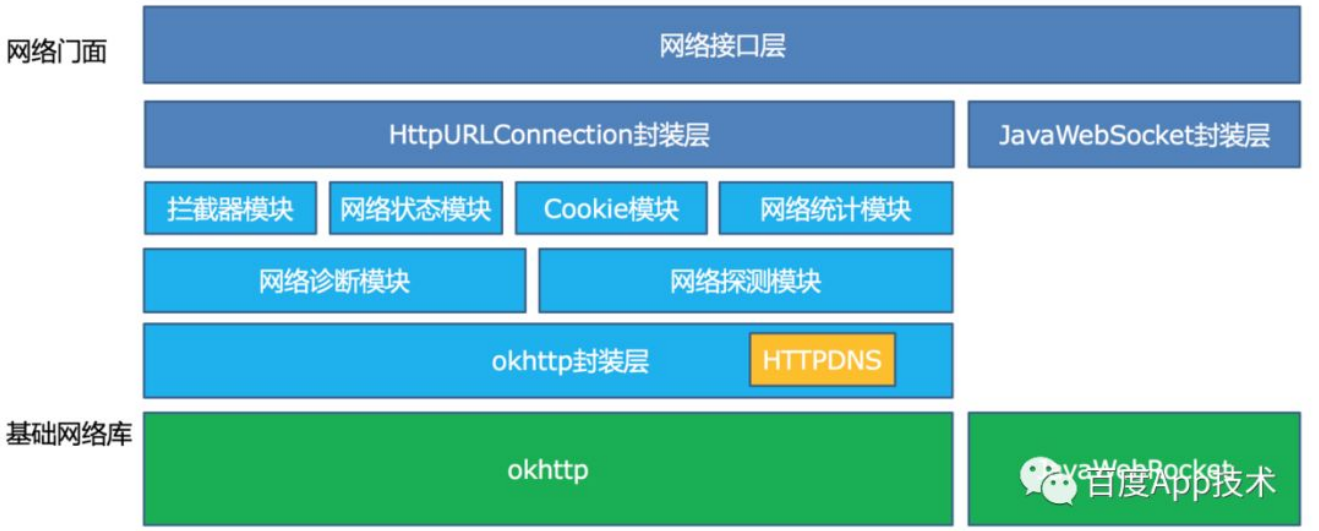
HTTPDNS的缓存场景，我们是选其一还是都选择呢？百度App选择的是内存缓存，目的是防止我们自己的服务出现问题，运维同学在紧急情况下切换流量，如果做了磁盘缓存，会导致百度App在重启后也可能不可用，但这种问题会导致APP在冷启动期间，HTTPDNS结果未返回前，还是存在故障或者劫持的风险，综合评估来看可以接受，如果出现这种极端情况，影响的是冷启动阶段的一些请求，但只要HTTPDNS结果返回后便会恢复正常。

四、HTTPDNS的最佳实践

百度App目前客户端网络架构由于历史原因还未统一，不过我们正朝着这个目标努力，下面着重介绍下HTTPDNS在Android和iOS网络架构中的位置及实践。

HTTPDNS在Android网络架构的位置及实践

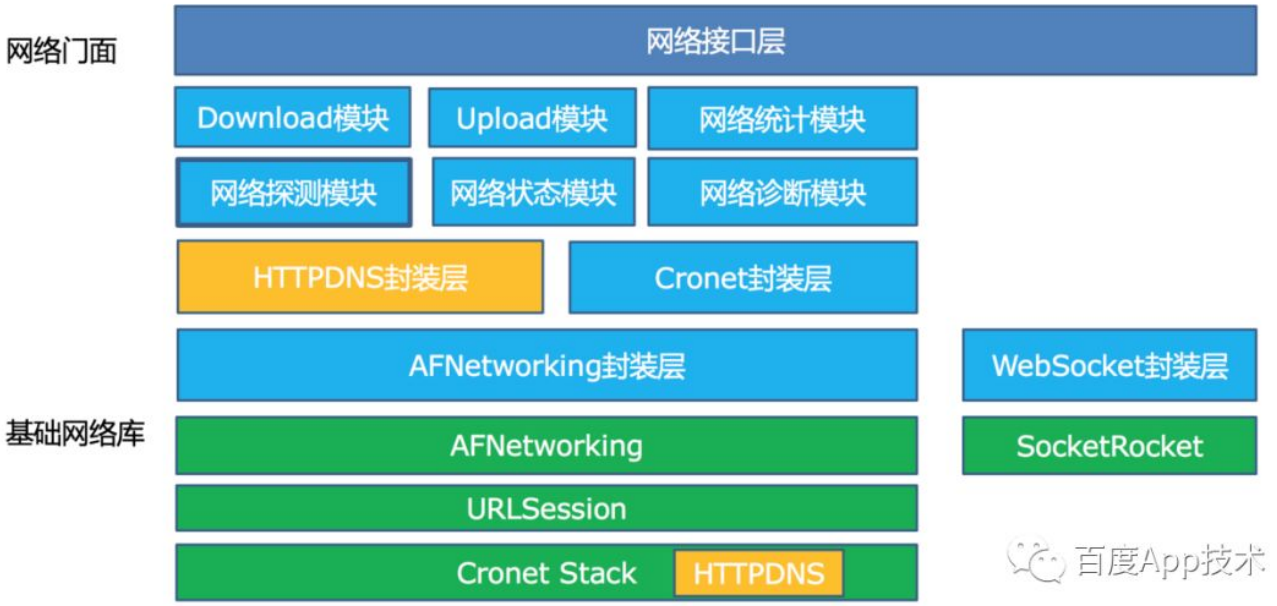
百度App的Android网络流量都在okhttp之上，上层进行了网络门面的封装，封装内部的实现细节和对外友好的API，供各个业务和基础模块使用，在okhttp上我们扩展了DNS模块，使用HTTPDNS替换了原有的系统DNS。



HTTPDNS在Android网络架构的位置

HTTPDNS在iOS网络架构的位置及实践

百度App的iOS网络流量都在cronet（chromium的net模块）之上，上层我们使用AOP的方式将cronet stack注入进URLSession里，这样我们就可以直接使用URLSession的API进行网络的操作而且更易于系统维护，在上层封装了网络门面，供各个业务和基础模块使用，在cronet内部我们修改了DNS模块，除了原有的系统DNS逻辑外，还添加了HTTPDNS的逻辑。iOS上还有一部分流量是在原生URLSession上，主要是有些第三方业务没有使用cronet但还想单独使用HTTPDNS的能力，所以就有了下面的HTTPDNS封装层，方法是在上层直接将域名替换成IP，域名对于底层很多机制是至关重要的，比如https校验，cookie，重定向，SNI（Server Name Indication）等，所以将域名修改成了IP直连后，我们又处理了以上三种情况，保证请求的可用性。



HTTPDNS在iOS网络架构的位置

五、收益

DNS优化的收益主要有两点，一是防止DNS的劫持（在出问题时显得尤为重要），降低网络时延（在调度不准确的情况下，会增大网络的时延，降低用户的体验），这两点收益需要结合业务来说，以百度App Feed业务为例，第一点上我们取得了比较大的效果，iOS劫持率由0.12%降低到0.0002%，Android劫持率由0.25%降低到0.05%，第二点的收益不明显，原因在于Feed业务主要目标群体在国内，百度在国内节点布局相对丰富，服务整体质量也较高，即使出现调度不准确的情况，差值也不会太大，但如果在国外情况可能会差很多。

六、结语

DNS优化是个持续性的话题，上面介绍的百度App的一些

经验和做法并不见得完美，但我们会持续深入的优化下去，为百度App的DNS能力保驾护航。最后感谢大家的辛苦阅读，希望对你有所帮助，后面会继续推出-百度App网络深度优化系列《二》连接优化，敬请期待。

七、个人心得

做为一个工程师，如何才能做好网络优化这件事情，是个值得我们交流探讨的话题，个人认为应该从以下五方面入手。

【1】基础知识要了解学习，要夯实，网络相关的内容很多，很杂，不易学习，啃过IETF发布的RFC的同学应该深有感触。

【2】学会将看不见的网络变成看得见的，很多自认为对于网络很了解的同学，动不动就背诵tcp协议原理，拥塞控制算法，滑动窗口大小等，但真正遇到线上问题，无从下手。对于客户端同学，我们在PC上要学会使用tcpdump和Wireshark等工具，适当使用Fiddler和Charles等工具，很多时候电脑和手机的网络环境不见得一致，所以要在手机上使用iNetTools，Ping&DNS或终端工具。学会使用工具后，要学着创造不同的网络环境，有很多工具能帮助你完成这点，比如苹果的Network Link Conditioner，FaceBook的ATC（Augmented Traffic Control）等。具备以上两个场景后，你的第一条储备就发挥了作用，你要能看懂握手过程，传输过程，异常断开过程等。

【3】有了以上两点的准备，接下来需要一个会出现各种网络问题的平台，给你积累经验，让一个个高压下的线上

问题锤炼你，折磨你。

【4】网络优化是需要数据支撑的，但数据的采集和分析是需要经验的，有些数据一眼看下去就是不靠谱的，有些数据怎么分析都是负向收益的，一般来说是有三重奏来对数据进行分析的，一，线下数据的采集和分析，得出正向收益，二，灰度数据的采集和分析，得出正向收益，三，线上数据的采集和分析，得出正向收益。

【5】数据的正向收益，不能完全证明提升了用户的体验，所以很多时候需要针对特定场景，特定case来分析和优化，就算是大家公认做的很好的微信，也不是在所有场景下都能保证体验上的最佳。

八、参考资料

- https://chromium.googlesource.com/chromium/src/+/HEAD/docs/android_build_instructions.md
- https://chromium.googlesource.com/chromium/src/+/HEAD/docs/ios/build_instructions.md
- https://github.com/Tencent/mars
- https://tools.ietf.org/html/rfc7858
- https://tools.ietf.org/html/rfc6555
- https://tools.ietf.org/html/rfc8305