百度App网络深度优化系列《三》弱网优化

一、前言

网络优化解决的核心问题有三个，**第一是安全问题**，我们在系列《一》DNS优化进行了详细的讲解。**第二是速度问题**，我们在系列《二》连接优化也做了详细的介绍。**第三是弱网问题**，它是网络优化中最为复杂且需要反复验证和分析的问题，我们的系列《三》弱网优化就是要深入探讨这个问题。

二、背景

弱网优化需要解决的核心问题有两点

【1】移动网络环境如此复杂，我们如何确定当下就是弱网环境。

【2】确定为弱网环境下，我们如何提升弱网下的成功率，降低弱网下的时延，进而提升用户的网络体验。

百度App承载着亿级流量，**弱网比例0.95%**，可谓不小，这个比例是如何得来的呢？还是要从什么是判断弱网指标说起。

三、判断弱网的指标

首先我们来探讨下都有哪些指标会影响到网络的质量，包

括httprtt，tcprtt，throughput，signal strength，bandwidth-delay product。

1.httprtt

httprtt（http Round-Trip Time）又名TTFB（Time to first byte），**指从客户端请求的第一个字节开始发送到接收到http header的第一个字节的时间差**。httprtt的时间如果过长，一方面是客户端本身接入网络质量的问题，另一方面是服务的延时比较大。

2.tcprtt

tcprtt（tcp Round-Trip Time）**指客户端tcp信道第一个字节发送到接收第一个字节的时间差**。因为HTTP协议底层是基于TCP的，所以在复用同一条tcp连接的前提下，httprtt的时间是包含tcprtt的时间的。大部分情况下httprtt已经可以说明问题的原因。

3.throughput

throughput，中文名字吞吐量，**它是用来衡量单位时间内成功传送数据的数量，是可以比较客观的衡量网络质量的指标**。吞吐量 =（获bits结束大小 - 获bits开始大小）/（获bits结束时间 - 获bits开始时间），这里有个细节需要注意，posix socket的read函数返回值是bytes，所以要乘以8得到bits。通常在httprtt比较小的情况下，网络依然很慢，这个时候就可以使用吞吐量来确定网络的质量。

4.signal strength

signal strength，**这里指的是无线信号强度**，在Android上可以通过PhoneStateListener的onSignalStrengthsChanged方法获取到信号强弱，但要注意只能在Android M以上的版本才生效。iOS上暂时没有靠谱的实现。

5.bandwidth-delay product

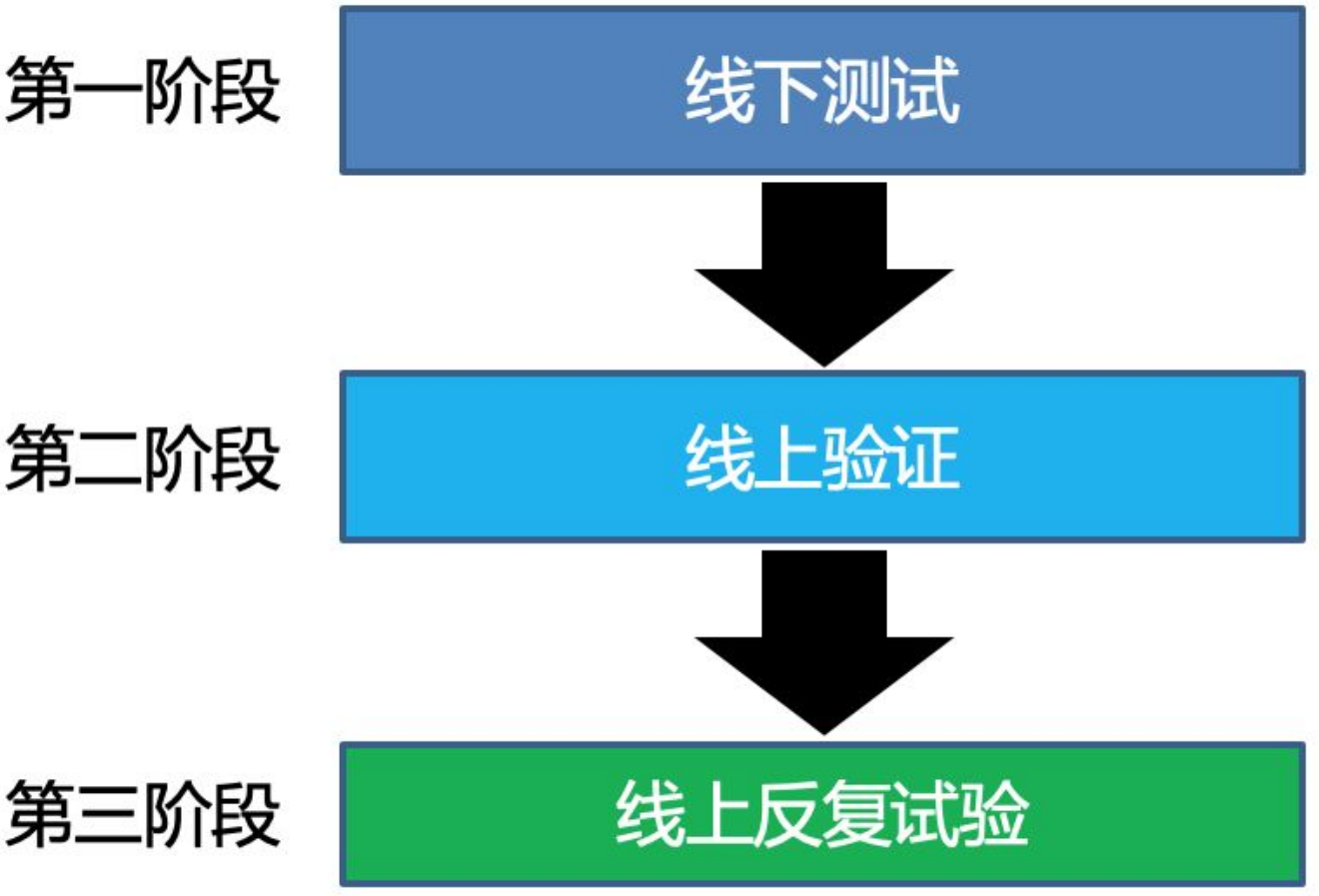
bandwidth-delay product，中文名**带宽时延乘积，指的是一个数据链路的能力（throughput）与来回通信延迟（rtt）的乘积**。带宽时延乘积的结果是比特不是位，这个比特值反应出当前网络管道的最大容量。TCP中有一个窗口大小的概念，会限制发送和接收数据的大小，所以TCP窗口大小的调节是直接受带宽时延乘积的影响，根据带宽时延乘积的值去设置套接字的setsockopt方法，设置的option是SO_RCVBUF（接收缓冲区大小）和SO_SNDBUF（发送缓冲区大小）。

通过上面的内容，我们对影响网络质量的指标有了一定了解，**对于不同的产品，影响网络质量的指标可以理解成一样的，但对于每个指标的阈值肯定是不一样的**，因为这包含着业务场景，比如抖音是视频类网络传输，微信是长连接数据传输，百度是文本图片类数据传输。还包括服务端配备，不同产品线的服务集群能力肯定不一样，比如返回客户端的服务端耗时肯定不一样。所以针对不同的产品弱网指标是基本一致的，但是指标的取值肯定是不一样的。

四、如何建立弱网标准

建立弱网标准是一个循序渐进的过程，在一穷二白的时候

我们应该如何建立这个标准呢？答案分为三个阶段。



建立弱网标准的步骤

1.**第一阶段，线下进行测试。**获取一些符合我们预期的阈值，这个时候我们需要借助一些网络测试工具，比如苹果的Network Link Conditioner，Facebook的ATC（Augmented Traffic Control），来获取到线下不同网络情况的阈值，一般我们会测试App冷启动的场景，网络切换的场景，DNS故障场景，弱网场景（一般都是配置上下行的带宽，丢包率，延迟，DNS延迟参数，或者更为简单的是使用工具默认的一些弱网配置）。

2.**第二阶段，线上进行验证。**通过线下充分测试获取到的阈值，在线上可以获取到弱网的比例，在这里百度App是针对特定场景的，比如Feed刷新，搜索落地页打开等，就算是在移动时代被大家公认的网络体验好的微信，也只是

在信令传输（收发消息）上做到极致优化，所以针对场景搜集弱网数据很重要。

3.**第三阶段，线上的反复试验**。想做到理想的弱网效果，少不了线上反复的阈值调整，通过调整阈值比较针对场景的网络请求的成功率、耗时、连接复用率等指标，使我们获得趋向于针对场景的合理阈值。

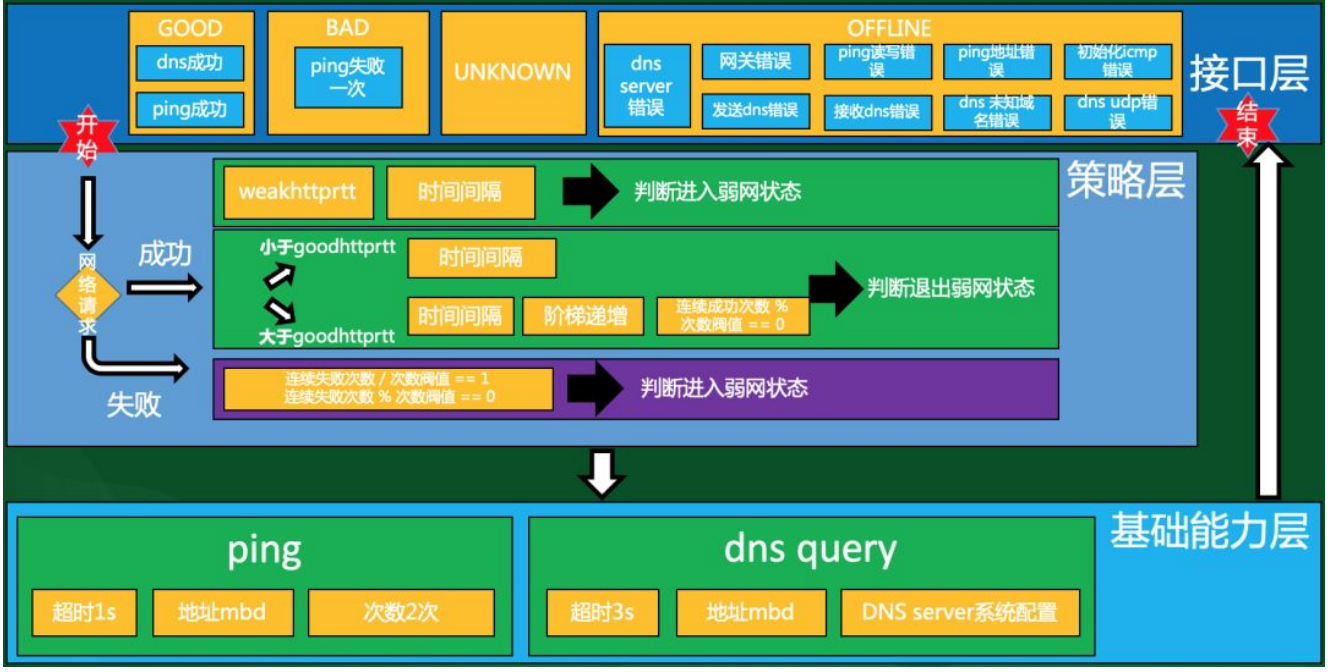
聊了这么多，那么弱网的探测如何实现呢?

五、网络探测的整体架构和实现

网络探测是弱网检测的基础，是否能即时，正确的检测出网络质量，是我们首先要解决的问题。**我们把网络探测划分为两部分，主动网络探测**和**被动网络采集**。

1.主动网络探测

所谓主动探测，**就是在触发了某些条件后，主动的进行网络探测，并按照一定的条件检查出是否是弱网状态**。百度App自研了主动探测组件，如下图所示。



主动网络探测

1.1策略层

探测策略层通过多种策略的组合，使主动探测的即时性和准确性得以大大提高，我们结合上面的策略层图来解释下检测维度的意义。

我们分别在网络请求成功和失败的时候触发了弱网检测的逻辑。主要分为如下三种逻辑。

1）成功时，如何判断进入弱网状态？检查weakhttprtt的阈值，这个值取决于业务的设置（**一般这个值会针对特殊场景的请求取95分位或者更大分位的值**），大于这个值就会进入弱网检测，为了防止频繁触发探测加了时间间隔维度，目前定义的是10分钟。从线下模拟测试来看，只要大于这个阈值，检测结果必然是弱网状态。

2）成功时，如何判断退出弱网状态？检查goodhttprtt的阈值，这个值取决于业务的设置（**一般这个值会取整体网络的95分位或者更大分位的值**），小于这个值证明要切换

回正常网络状态，为了防止频繁触发探测加了时间间隔维度，目前定义的是30秒。从线下模拟测试来看，只要小于这个阈值，检测结果必然是正常状态。如果大于或者等于这个阈值，也不能证明一定不是正常网络，所以也需要发起网络探测，但是由于这是在成功回调里，频次会很高，所以我们加上时间间隔的限制30秒，还加入了次数的限制，连续成功次数%次数阈值（4次）等于0。但这看起来还是频次有点高，所以我们引入了阶梯递增机制，随着次数的增长，成60秒几何倍数增长。

3）失败时，如何判断进入弱网状态？首先会判断连续失败次数，连续失败次数/次数阈值（2次）等于1并且连续失败次数%次数阈值（2次）等于0，**相比成功，失败的次数检查较为苛刻，主要还是考虑多次触发网络检测损耗性能**。

进入弱网状态后，就会触发基础能力层的ping和dns query的探测。

1.2基础能力层

探测基础能力层，主要提供弱网检测的手段，**一是dns query，一是ping**，百度App使用C++实现了这两个能力。为什么要选用这两种手段呢？我们在系列二中介绍过，一个网络请求，分为DNS-》TLS-》TCP-》数据传输四个阶段。想判定网络连通性主要在DNS和TCP阶段，所以dns query和ping就是用来检测这两个阶段的连通性手段。dns query向百度核心域名mbd.baidu.com发起dns查询，查询的DNS服务器为系统配置的DNS服务器（iOS通过res_ninit函数构建一个__res_state的结构体，Android通

过systemproperty获取net.dns1和net.dns2的值，便可获取系统配置的DNS服务器），DNS查询的超时时间为3s。ping的目标地址为百度核心域名mbd.baidu.com，ping的次数为两次，每次超时时间是默认的1s。

判断出弱网状态后，会将结果提供给接口层。

1.3接口层

接口层主要提供主动探测出来的网络状态，**目前包括GOOD，BAD，UNKNOWN，OFFLINE**。

1）GOOD：dns查询成功并且ping也成功，即标记为GOOD状态。

2）BAD：ping失败一次标记为BAD状态。

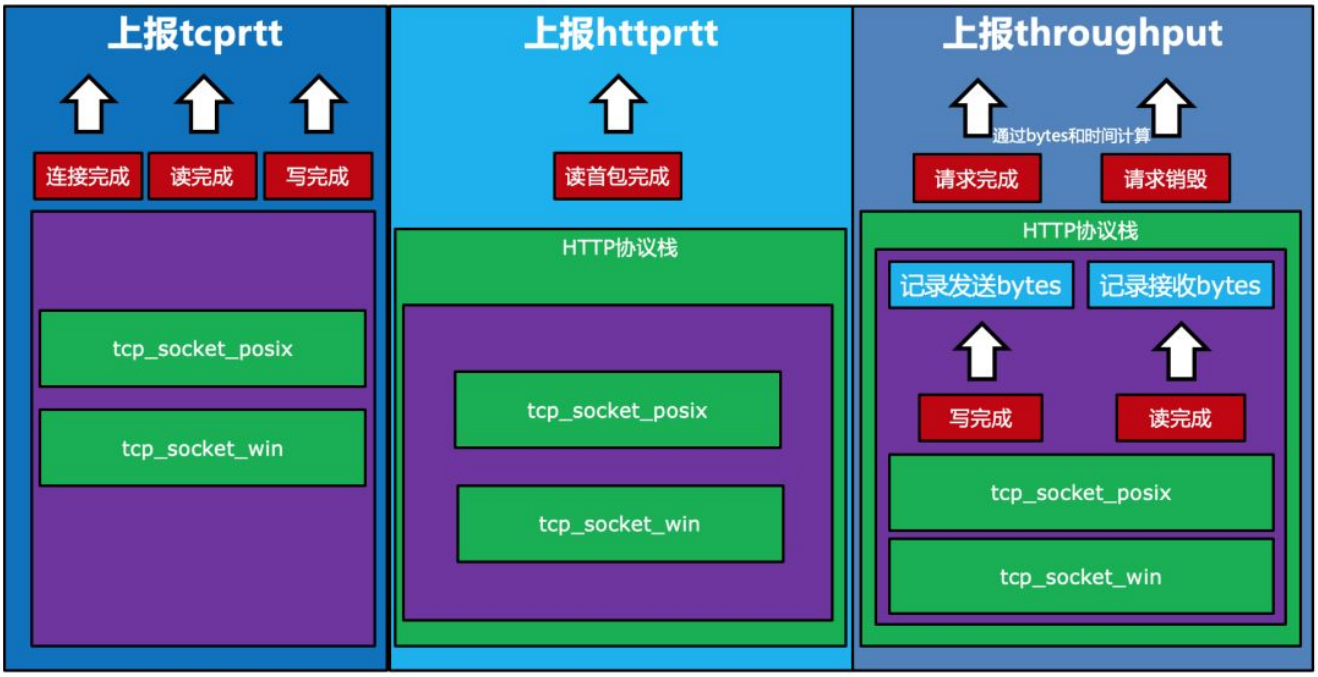
3）UNKNOWN：初始状态或者识别不出来状态为UNKNOWN状态。

4）OFFLINE：dns server错误（没有获取到要发送的DNS server地址），网关错误（读取/proc/net/route文件内容失败），发送dns错误（发送dns数据出错），ping读写错误（ping的过程中读写错误），接收dns错误（接收dns数据出错），ping地址错误（ping地址是空），dns未知域名错误（dns没有查询到域名错误），初始化icmp错误（初始化icmp失败），dns udp错误（创建UDP socket失败），即标记为OFFLINE状态。

2.被动网络采集

所谓被动采集，**就是每一次网络请求的所有细节都进行记录，并按照一定的条件将原始信息进行上报，上层再根据条件判断是否是弱网状态。**百度App基于cronet的NQE（Network Quality Estimator）进行了二次订制开发。

首先我们讲解下需要采集的数据，包括tcprtt、httprtt、throughput三个维度，如下图所示。



被动采集数据

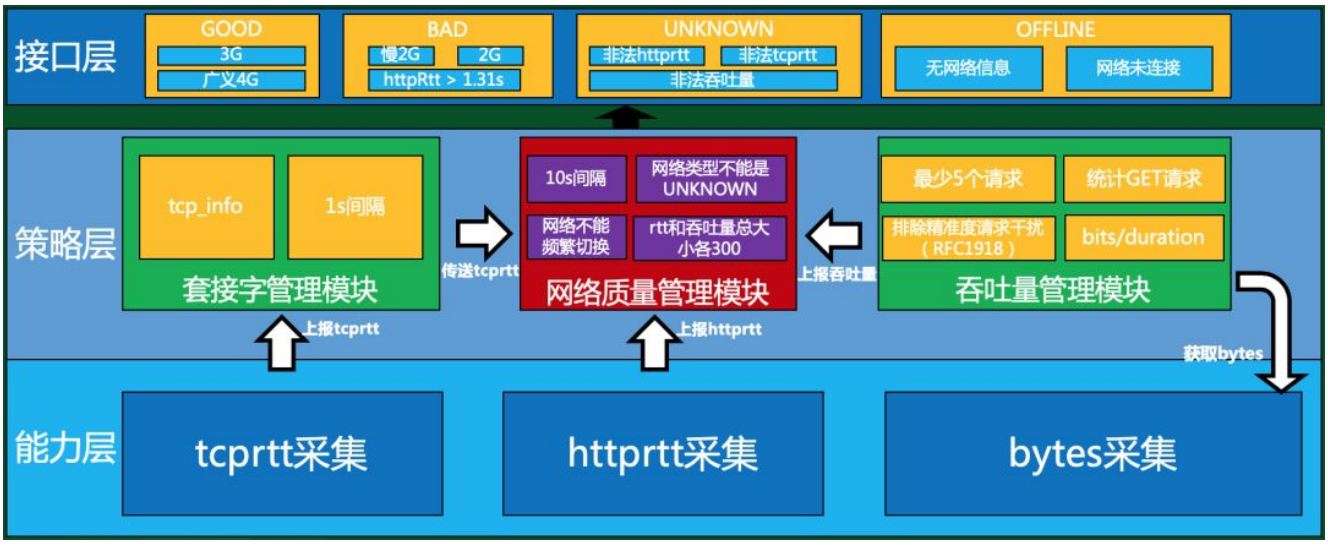
1）tcprtt，基于posix和windows的socket编程接口来获取tcprtt。获取时机在连接完成，读完成和写完成。

2）httprtt，基于HTTP协议栈实现，通过计算接收response数据开始和开始发送的时间差，来获取httprtt。获取时机在读首包完成时。

3）throughput，通过上面的计算公式需要获取bytes和时间，基于posix和windows的socket编程接口来获取bytes。获取时机在读完成时记录接收的bytes，在写完成

时记录发送的bytes。时间的获取在吞吐量管理模块里完成，下面会讲到。获取时机在请求完成和请求销毁时。

如下为被动网络采集的整体架构图。



被动网络采集

1.1能力层

能力层内容上面我们已经讲过，主要采集tcprtt、httprtt、throughput三个维度的数据。

1.2策略层

被动采集策略层通过多种策略的组合，降低各种采集数据的上报时机，降低性能的影响。

1）套接字管理模块，首先负责获取tcprtt的值，如何获取tcprtt呢？通过getsockopt函数获取tcp_info结构体里的tcpi_rtt值。其次由于tcprtt的上报频次比较频繁，所以做了1秒的时间间隔上报限制。

2）吞吐量管理模块，负责吞吐量的计算，上面介绍了计算公式，从网络活动监控器模块获取bytes，但吞吐量的

计算单位是bits（位），所以将bytes乘以8。只有GET请求会被列入统计计算，并且至少要累计5个请求后才能开始统计计算。排除精准度的干扰，比如localhost，私有子网上的主机，特定用途子网主机，可参考RFC1918。

3）网络质量管理模块，从套接字管理模块获取tcprtt，从吞吐量管理模块获取吞吐量，并且在HTTP协议栈读首包完成时获取httprtt。获取到这三个值后，需要经过一些策略限制上报的频次，10秒间隔的限制；网络类型不能是UNKNOWN（1.3的第三部分会详细讲解）；网络不能频繁切换；rtt和吞吐量总大小各300个。

1.3接口层

接口层主要提供被动采集出来的网络状态，目前包括GOOD，BAD，UNKNOWN，OFFLINE。

1）GOOD：3G和广义的4G，任一条件满足标记为GOOD状态。通过阈值标记3G和广义的4G，httprtt大于等于273ms，tcprtt大于等于204ms，即标记为3G状态。小于这两个值被标记为广义的4G，所谓广义的4G包含4G、WiFi、以及质量较好的各种接入网络。

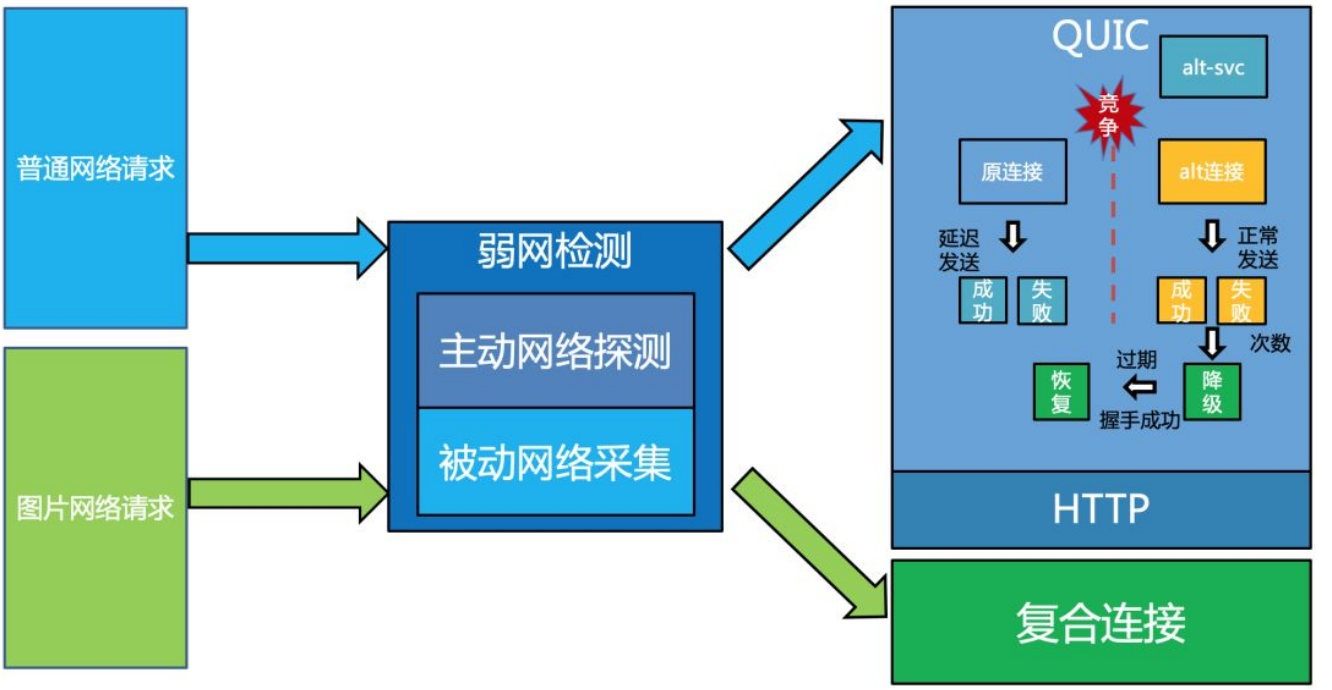
2）BAD：慢2G，2G和httprtt大于1.31秒，任一条件满足标记为BAD状态。通过阈值标记慢2G和2G，httprtt大于等于2.01秒，tcprtt大于等于1.87秒，标记为慢2G。httprtt大于等于1.42秒，tcprtt大于等于1.28秒，标记为2G。httprtt大于1.31秒，为百度App的Feed刷新业务阈值。

注：上面涉及的时间值为nqe内部的机制，具有普适性。

3）UNKNOWN：非法的httprtt，tcprtt，吞吐量，任一条件满足标记为UNKNOWN状态。何为非法？值为-1为非法，那什么条件被标记为-1呢？首先初始化时会被标记为-1，其次在从来没有获取到过httprtt，tcprtt，throughput的值时，会使用本地默认的值做为判断标准，这是一种容错处理。

4）OFFLINE：依赖平台能力进行判断，Android平台依赖ConnectivityManager获取NetworkInfo，通过NetworkInfo的isConnected获取是否连接，如果未连接则判断为OFFLINE状态，如果NetworkInfo为空则判断为OFFLINE状态。

六、弱网状态下百度App如何改善用户体验



百度App在弱网下的手段

1.QUIC在百度App弱网下的最佳实践

QUIC（Quick UDP Internet Connections）是新一代的互联网传输协议，最早源于Google，它的详尽内容可参考资料【3】，本章我们不做QUIC的科普介绍。

百度App的普通网络请求在弱网状态下会切换到QUIC，本章重点讲解下百度App针对弱网下开启QUIC后遇到的问题，**一是开启QUIC一旦遇到问题是否可以回滚？二是在弱网下如何能让流量尽可能的走QUIC？针对这两个问题，我们的解决方案是QUIC升降级原理和QUIC预连接。**

1.1QUIC升降级原理

如上图QUIC部分所示，QUIC的升降级依赖于HTTP Alternative Services，HTTP Alternative Services是与HTTP有关的一个协议，它不是为QUIC专门设计的，在HTTP协议上主要负责新服务的替换，对于HTTP1.1协议，它是通过HTTP响应头传输回来的，所以只能在第二次请求时生效，如下面格式。

Alt-Svc: quic="alt.example.com:443", quic=":443"; ma=2592000

如上信息表明切换到quic协议，指定了域名服务和端口，并且指定了生效时间，以秒为单位。

Alt-Svc: clear

如上信息表明将alter配置清空

在网络库内部有一个alter连接和原连接的竞争机制，如果alter信息已经存在，优先发送alter连接，而原连接会延迟发送，延迟时间默认300ms，谁先成功就使用哪个连接，

如果alter连接在QUIC握手时失败，会记录这个alter信息的失败次数，并根据失败的次数，计算出一个过期时间，这个过期时间会随失败次数指数增加，最长为2天。当过期时间到期后，会清除这个alter信息，当这个alter连接在QUIC握手成功后，会清除这个alter信息。

1.2QUIC预连接

所谓QUIC的预连接，就是在进入弱网状态前提前建立QUIC连接。大家都知道QUIC引以为傲的0RTT，但第一次建立连接的时候是需要1RTT的，客户端首先会向服务器发送一个client hello消息，服务器会回复一个server reject消息，这个消息中包括了server config，有了server config后客户端就可以直接计算出密钥，完成0RTT，详尽内容可参考资料【4】。

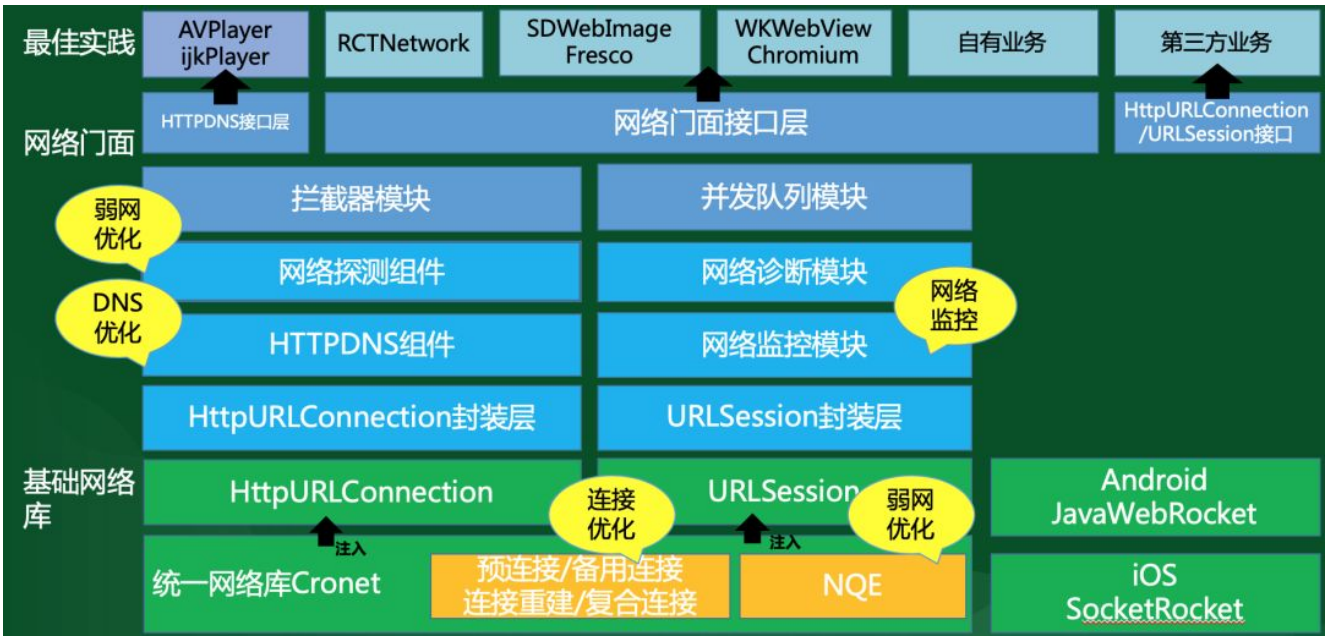
通过上面的原理，客户端拉取server config的成功概率会直接影响QUIC在弱网下的流量，所以我们在App启动的过程中会做一次QUIC预连接，将server config拉取下来，这样等进入弱网后alter连接会大概率的竞争过原连接，进而走QUIC协议。

2.复合连接在百度App弱网下的最佳实践

复合连接的具体原理可以查看《百度App网络深度优化系列《二》连接优化》里的具体介绍，百度App目前在弱网下只是针对图片网络请求开启了复合连接，因为图片请求不管是HTTPDNS结果还是localDNS的结果都是多个IP，这是满足复合连接的前提。在弱网下多IP的尝试会比单IP的结果好些，另外弱网的比例相对较小，复合连接对于服

务器的负载也会小些。

七、百度App网络整体架构



百度App网络整体架构

百度App网络整体架构，**以网络门面为中间层，隔离上层的最佳实践和底层的基础网络库。**

1.1最佳实践

客户端网络库的一部分工作量是在如何让最佳实践做的更好，在音视频上，不管是iOS的AVPlayer，还是双端的ijkPlayer，都是使用HTTPDNS组件接管DNS模块，没有全部接管网络模块。ReactNative的网络模块RCTNetworking，图片库Android的Fresco和iOS的SDWebImage，WebView组件Android的Chromium和iOS的WKWebView，以及百度App的自有业务，都是通过网络门面的接口层直接接管。而对于第三方业务考虑到与宿主耦合的关系，直接使用Android的HttpURLConnection和iOS的URLSession系统标准的接口。

1.2网络门面

网络门面主要包括，拦截器模块（提供给业务订制网络门面的机制）、并发队列模块（提供高中低以及非常高的网络请求优先级）、网络探测组件（弱网主动探测能力）、网络诊断模块（包括https，ping，dns的校验）、HTTPDNS组件（《百度App网络深度优化系列《一》DNS优化》里详细讲解）、网络监控模块（客户端的打点机制，服务端的例行和突发监控）、HttpURLConnection封装层、URLSession封装层。

1.3基础网络库

基础网络库包含两部分，一部分是基于cronet二次订制的统一网络库，一部分是WebSocket基础库（Android的JavaWebSocket，iOS的SocketRocket）。统一网络库内部包含连接优化的内容（在《百度App网络深度优化系列《二》连接优化》里详细讲解），弱网优化的内容（上面提到的被动采集）。通过AOP的方式将底层协议栈注入进HttpURLConnection（利用URLStreamHandlerFactory）和URLSession（利用URLSessionConfiguration的protocolClasses属性），两者都是系统提供的URL Loading机制。

八、收益

弱网优化的收益我们主要从上面讲到的进入弱网状态后的手段来看，包括开启QUIC，QUIC预连接，开启复合连接。

1）弱网下开启QUIC后，网络连接成功率提升0.01%，平均耗时降低23.5%。

2）弱网下开启QUIC预连接后，QUIC协议的pv从37万涨到90万。

3）弱网下开启复合连接后，bad状态下耗时降低2.5%，offline状态下耗时降低7.7%。

九、结语

系列一到系列三的内容到今天全部完成，希望能对大家的工作和学习有所帮助，感谢大家的持续关注和鼓励。**生命不息，优化不止，做技术我们是认真的。**

十、参考资料

1. https://chromium.googlesource.com/chromium/src/+/HEAD/docs/android_build_instructions.md

2. https://chromium.googlesource.com/chromium/src/+/HEAD/docs/ios/build_instructions.md

3. https://www.wolfcstech.com/2019/03/27/quic_2019_03_27/

4. https://www.wolfcstech.com/2017/03/09/QUIC%E5%8A%A0%E5%AF%86%E5%8D%8F%E8%AE%AE/

5. https://tools.ietf.org/html/rfc1918

6. https://github.com/Tencent/mars