百度App网络深度优化系列《二》连接优化

一、前言

在系列《一》里大家了解到网络优化一般会**首选优化DNS**，而接下来的HTTP协议成为优化的重点，一般优化者会选择协议切换，合并请求，精简数据包大小等手段来对HTTP协议进行优化，严谨的说这都不属于网络优化的范畴。

HTTP协议的基础是连接，所以我们的系列《二》连接优化应运而生，希望对大家在网络方向的学习和实践有所帮助。

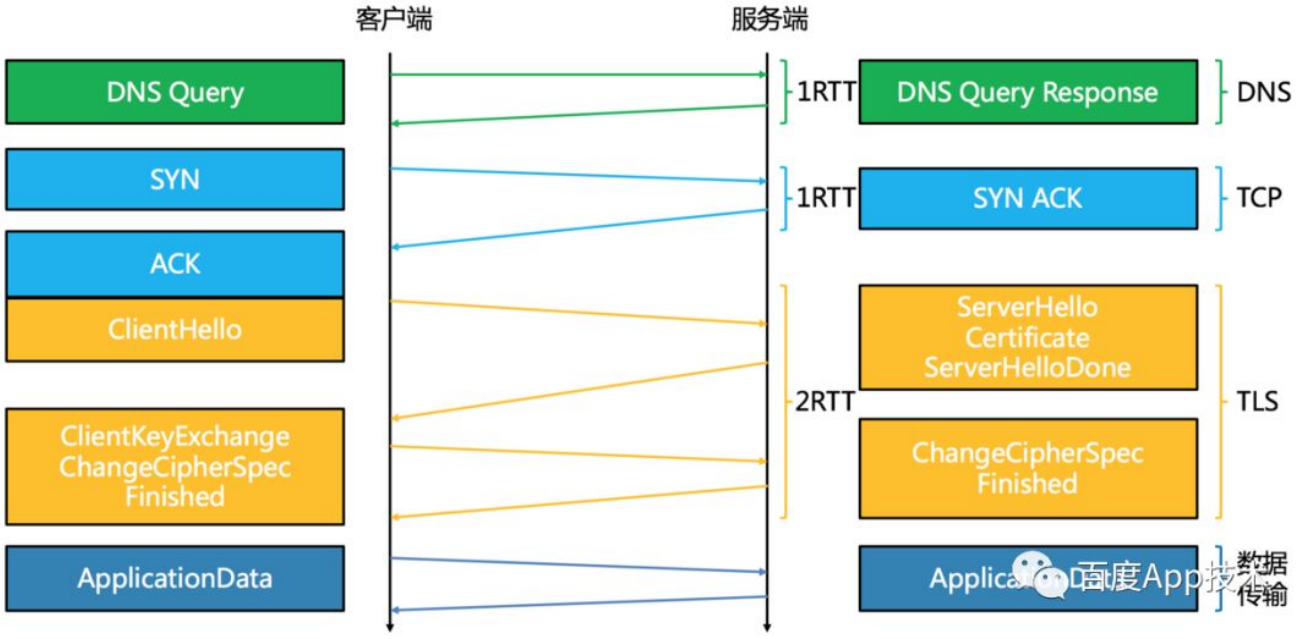
二、背景

连接优化需要解决两个核心问题

1. 连接建立耗时较长，导致请求的总时长变长，进而影响用户体验。

2. 在多变的网络环境下，连接建立的过程可能会失败，导致成功率下降，进而影响用户体验。

百度App承载着亿级流量，对于每一个请求都需要追求耗时短，成功率高的体验。从协议角度出发，如何才能做到这一点呢？首先我们来看下建立连接耗时的原理。



建立连接耗时的原理

从上图我们能清晰的看出

1. DNS Query需要1个RTT（Round-Trip Time，即往返时间），百度App都是基于HTTPDNS服务的，所以大部分会命中缓存，如果降级走了系统DNS，也会命中缓存，命中不了的由于是基于UDP协议，所以在连接耗时上没有太大的影响，线上的数据也能说明这点。

2. TCP要经历SYN，SYN/ACK，ACK三次握手的1.5个RTT，不过ACK和ClientHello合并了，所以就是1个RTT。

3. TLS（Transport Layer Security，即传输层安全性协议）需要经过握手和密钥交换2个RTT。

综上所述，DNS，TLS，TCP握手阶段用了4个RTT才到了ApplicationData阶段，也就是数据开始传输阶段。

通过上面的分析可以总结出，如果我们能尽量的将TLS和TCP的RTT减少，将会大大降低连接耗时的时间。

三、连接优化我们都能做什么

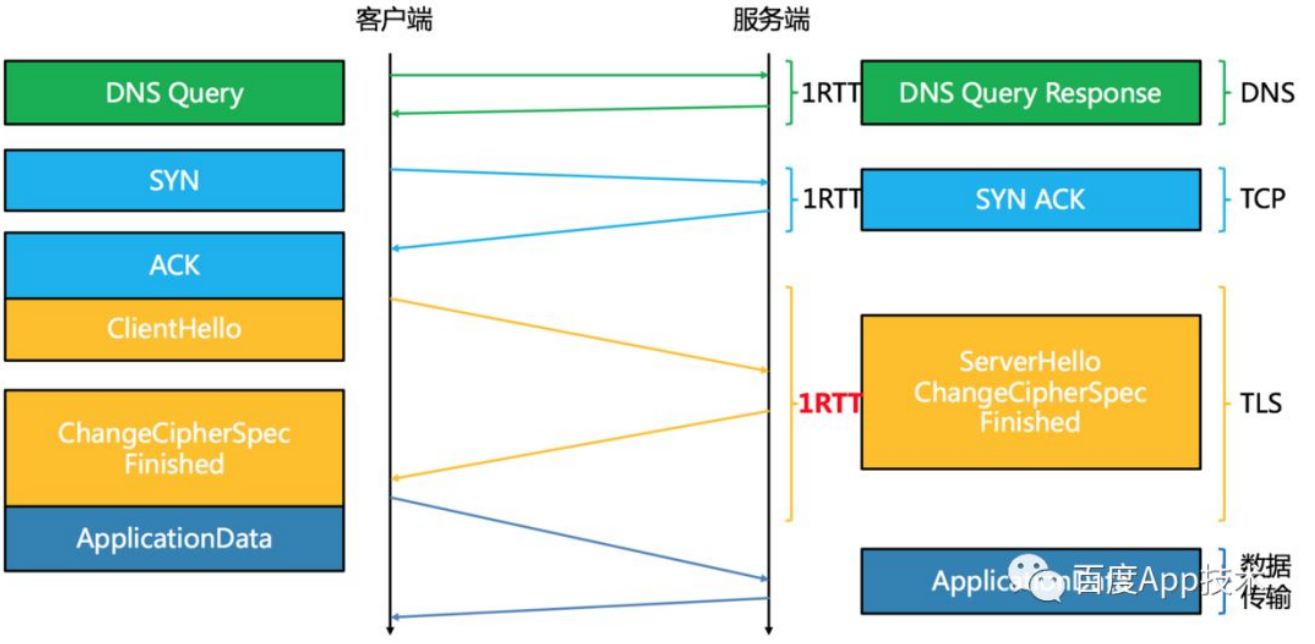
百度App的优化目标分为两类，**一类是TLS的连接优化，一类是TCP的连接优化。**

TLS的连接优化

TLS的连接优化，需要服务端和客户端都需要支持，共同完成优化手段，包括Session Resumption和False Start。

Session Resumption

Session Resumption中文意思是会话复用，下图讲解了Session Resumption的协议原理。



Session Resumption的协议原理

通过上图可以看出TLS密钥协商交换的过程没有了，但具体是如何实现的呢？包含两种方式，**一种是Sesssion Identifier，一种是Session Ticket。**

1）Session Identifier

Session Identifier中文为会话标识符，**更像我们熟知的session的概念**。是 TLS 握手中生成的 Session ID。服务端会将Session ID保存起来，客户端也会存储Session ID，在后续的ClientHello中带上它，服务端如果能找到匹配的信息，就可以完成一次快速握手。

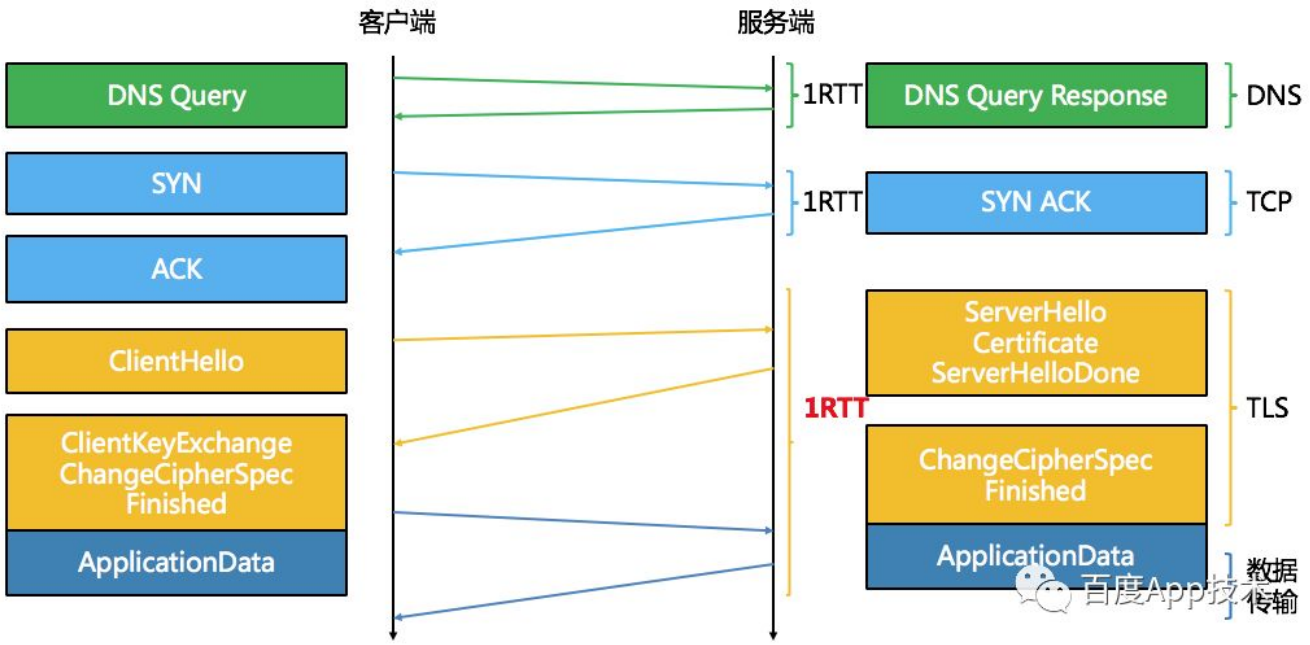
2）Session Ticket

Session Identifier存在一些弊端，比如客户端多次请求如果没有落在同一台机器上就无法找到匹配的信息，但Session Ticket可以。**Session Ticket更像我们熟知的cookie的概念**，Session Ticket用只有服务端知道的安全密钥加密过的会话信息，保存在客户端上。客户端在ClientHello时带上了Session Ticket，服务器如果能成功解密就可以完成快速握手。

不管是Session Identifier还是Session Ticket都存在时效性问题，不是永久生效，对于这两种方式大家可以查看参考资料【4】。**百度App的网络协议层对这两种方式都是支持的**，省去了TLS握手过程中证书下载，密钥协商交换的环节，节省了1个RTT的时间。

False Start

False Start的中文意思是抢跑，下图讲解了False Start的协议原理。



False Start的协议原理

上图很清晰的说明在TLS第一步握手成功后，客户端在发送Change Cipher Spec Finished的同时开始数据传输，服务端在TLS握手完成时直接返回应用数据。**应用数据的发送实际上并未等到握手全部完成，所以称之为抢跑。**

从结果看省去了1个RTT的时间。False Start有两个前提条件，一是要通过应用层协议协商ALPN（Application Layer Protocol Negotiation）握手，二是要支持前向安全的加密算法。False Start在未完成握手的情况下就发送了数据，前向安全可以提高安全性，具体协议实现，大家可以查看参考资料【3】。百度App的网络协议层对False Start是支持的。

这里说句题外话，其实TCP层有个类似的连接优化手段叫Fast Open，感兴趣的同学，可以查看参考资料【5】。

Session Resumption和False Start的区别

两者对于TLS来说都是节省一个RTT，Session

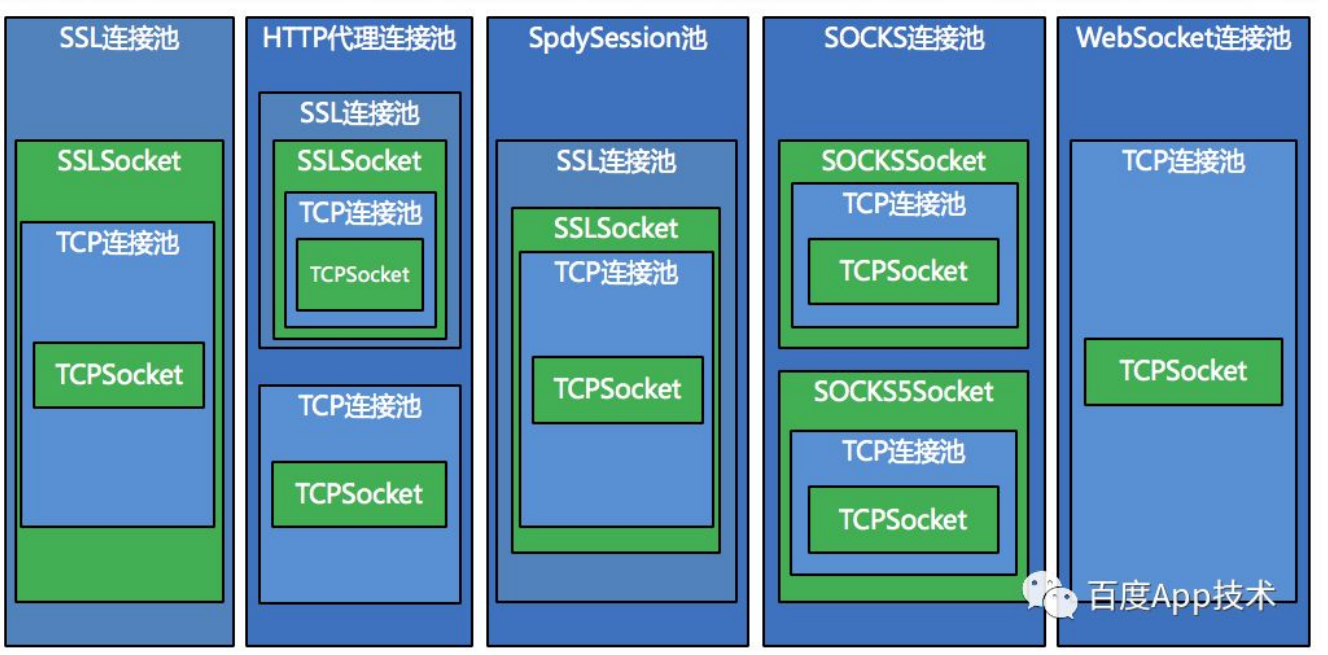
Resumption在第一次握手时还是需要2个RTT，在第二次握手时才能复用减少到1个RTT。False Start是端上的行为，故每次都会减少到1个RTT。

---此处内容较多 休息下再继续阅读吧---

TCP的连接优化

TCP的连接优化，我们先从连接池说起，首先让我们来认识下连接池都有哪些类型。

1. 连接池



连接池的类型

上图展示了连接池的不同类型，都是大家耳熟能详的协议连接池，有低级连接池，包含TCP连接池（管理HTTP请求的连接）和WebSocket连接池（管理WebSocket连接）。

有高级连接池，包括HTTP代理连接池（管理HTTP代理请求的连接），SpdySession连接池（管理SPDY和HTTP/2请求的连接），SOCKS连接池（管理SOCKS和SOCKS5代理的连接），SSL连接池（管理HTTPS请求的连接）。

不同类型的连接池以组合的形式互相复用能力。

1）SSL连接池管理的是SSLSocket，但SSLSocket又依赖于TCP连接池提供的TCPSocket。

2）HTTP代理连接池如果走HTTP协议，那么就需要TCP连接池提供TCPSocket，如果走HTTPS协议，那么就需要SSL连接池提供SSLSocket。

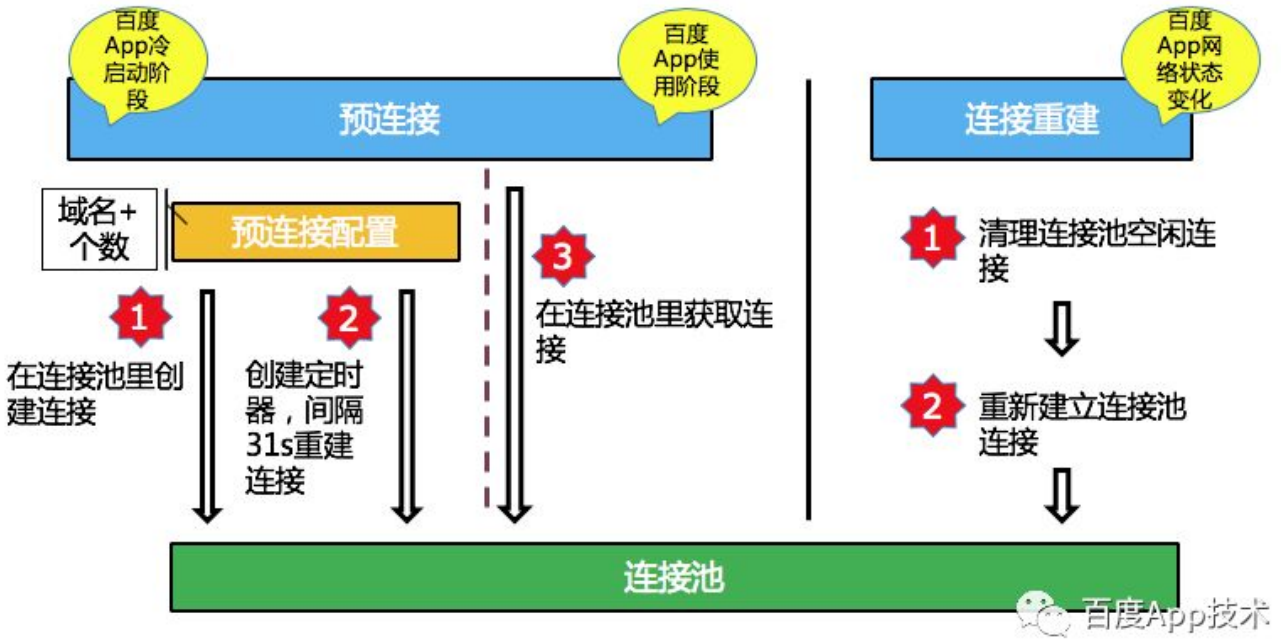
3）SpdySession连接池依赖SSL连接池提供SSLSocket，这里需要说明下，虽然HTTP/2协议没有强制绑定HTTPS，但是在实际开发中确实都是绑定HTTPS，百度App使用ALPN来协商HTTP/2。

4）SOCKS连接池管理的SOCKSSocket和SOCKS5Socket都需要依赖TCP连接池提供的TCPSocket，虽然SOCKS5支持UDP，但cronet网络库暂时没有实现。

5）WebSocket连接池依赖TCP连接池提供的TCPSocket，声明下这里没有说明WSS（Web Socket Secure）的情况。

TCP连接优化是一个比较复杂的内容，**百度App做了针对性场景优化，包括预连接，连接重建，备用连接，复合连接。**

2. 预连接



预连接和连接重建

预连接，预先创建好的连接。**它解决的场景是在App使用阶段可以无耗时的获取连接**。下面用四个问答来解释预连接。

问题一：预连接是否能解决所有网络请求的提前连接建立?

答：答案是否定的，预连接需要业务方进行核心业务的评估，针对核心的域名进行预连接的建立。

问题二：预连接既然针对的是特定的域名，那么是如何配置的呢?

答：采用域名+连接数的方式进行配置，比如https://a.baidu.com|2，表示给a.baidu.com这个域名配置两条预连接，这里要说明下，在HTTP/1.x协议下，网络库的实现都会对于单域名有最大连接数的限制，不同网络库

的个数限制不一样，有5个也有6个，但对于HTTP/2协议，这个连接数就只能是1个。

问题三：预连接是如何建立的?

答：在网络库初始化的时候，会根据使用者的配置延迟5s进行预连接的建立，主要是考虑网络库在冷启动下对于启动性能的影响，为了保证网络库的整体性能，预连接的总个数限制在20个。

问题四：预连接是如何保持的?

答：在网络库初始化的时候，除了进行预连接的建立，还会创建一个预连接的定时器，这个定时器会每隔31s，这个值的设定取决于BFE（Baidu Front End，是七层流量的统一接入系统）和BGW（Baidu Gate Way，百度自主研发的四层负载均衡平台）对超时的最小值设定，根据使用者的配置重新建立连接。

3. 连接重建

连接重建，将连接重新建立。**它解决的场景是App网络状态发生变化，IP地址变化，导致连接不可用**。下面用三个问答来解释连接重建。

问题一：连接重建是否针对连接池里的所有连接?

答：答案是肯定的。

问题二：连接重建的过程是什么样的?

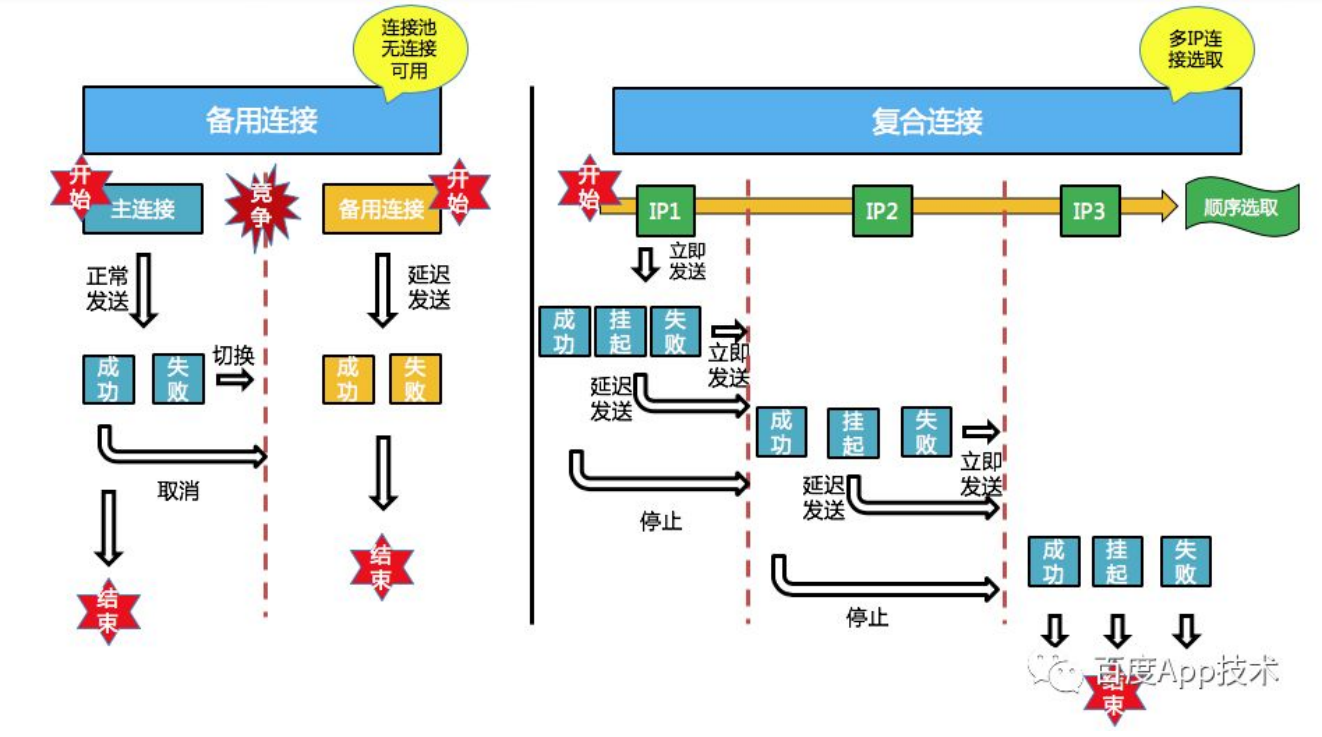
答：在网络状态变化的时候，第一步会清除掉连接池里的

idle socket，何为idle socket？即空闲socket，对于从未使用过的空闲socket超过60秒清除，对于使用过的空闲socket超过90秒清除。第二步重建连接需要等待200ms，目的是等待DNS先重建完成。

问题三：连接重建对于性能有影响吗？

答：出于性能考虑，连接重建的连接个数限制是100个。

4. 备用连接



备用连接和复合连接

备用连接，预备的连接。**它解决的场景是正常发送一个请求当group内无连接可用的时候**（何为group？group是管理socket的最小单元，内部包含活跃socket，空闲socket，连接任务，等待请求）。下面用三个问答来解释备用连接。

问题一：备用连接是否针对所有请求？

答：答案是肯定的。

问题二：备用连接的过程是什么样的?

答：当有请求来临时，连接池内无连接可用，会启动一个定时器开启备用连接，定时器的间隔时间是250ms，与主连接进行竞争，如果主连接因为网络抖动或者网络状态不好，导致连接失败，那么备用连接就直接发送请求。如果主连接成功，那么备用连接就被取消掉。

问题三：备用连接的目的是什么?

答：在连接池无连接的情况下，务必是要创建连接的，在主连接之外加一个备用连接，会大大提升创建连接的成功率，从而提升用户体验。

5. 复合连接：

复合连接，即多条连接。**它解决的场景是为了多个IP地址的连接选取问题**。下面用三个问答来解释复合连接。

问题一：复合连接是否针对所有请求?

答：答案是肯定的。复合连接可以全局开关，百度App现阶段暂时没有开启复合连接。

问题二：复合连接的过程是什么样的?

答：众所周知域名DNS查询一般情况下会返回多个IP，我们以域名查询返回两个IP为例

1）如果结果中存在IPv6的地址，那么会优先选用IPv6的

地址，这个规则follow HappyEyeBall机制（可参考系列一对于HappyEyeBall的介绍）。

2） 接下来这两个IP会按照顺序尝试建立连接，如果第一个IP返回失败，将立即开始连接第二个IP。

3）如果第一个IP率先成功返回，那么第二个IP将被加入连接尝试列表并停止所有尝试连接。

4）如果第一个IP失败，会立刻开始第二个IP的连接。

5）如果第一个IP处于pending状态，那么会启动一个定时器，默认延迟2s会发起第二个IP的连接，如果是多个IP将会递归连接，需要特别说明下，不同的网络制式延迟时间会不一样，这样体验也会更好。

问题三：复合连接的目的是什么？

答：复合连接的好处是提供最优的IP选取机制，但也会带来服务端的高负载，所以使用的时候需要进行综合评估。

四、连接优化的最佳实践

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。

我们的中心思想是以系统网络库的API调用接口为中心，上层建立网络门面，供外部便捷调用，底层通过系统机制以AOP的方式将cronet（chromium的net模块）注入进系统网路库，达到双端网络架构统一，能力复用。

下面着重介绍下连接优化在Android和iOS网络架构中的位

置及实践。

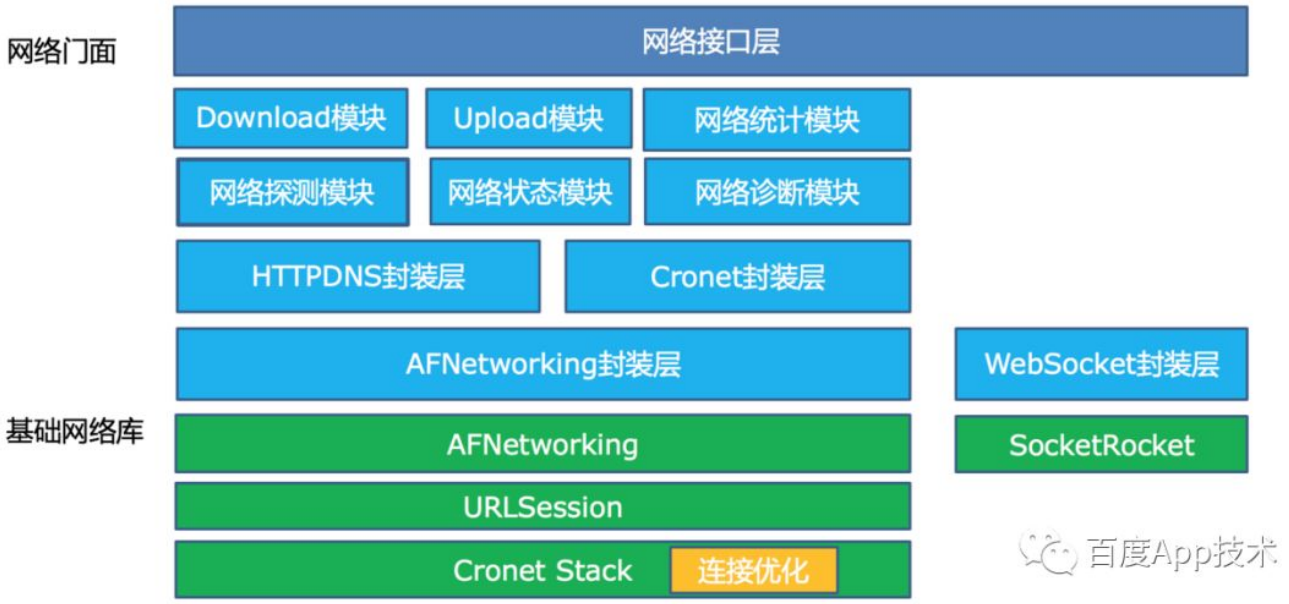
1. 连接优化在Android网络架构的位置及实践

连接优化在Android网络架构的位置

百度App的Android网络流量目前都在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，目前我们正在进行重构，默认采用Android标准的网络接口HttpURLConnection，它的底层由系统提供的okhttp的实现。

订制方面利用URL Stream Protocol机制将HttpURLConnection底层网络协议栈接管为cronet，供各个业务和基础模块使用，连接优化的所有内容在cronet网络库内部实现。

2. 连接优化在iOS网络架构的位置及实践



连接优化在iOS网络架构的位置

百度App的iOS网络流量目前都在cronet之上，上层我们使用iOS的URL Loading System机制将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内部实现了预连接（主要针对百度App的几个核心域名进行预连和保活），连接重建（针对所有请求），备用连接（针对所有请求），复合连接（iOS上暂时没有开启），Session Resumption（针对所有请求），False Start（针对所有请求）。

五、收益

连接优化的收益主要体现在网络时延和网络成功率上，这两点收益需要结合业务来说，以百度App Feed刷新这个典型业务场景为例。

Feed刷新文本请求网络时延**降低16%**，Feed刷新图片请

求网络时延**降低12%**，可谓收益相当明显。

成功率方面，Feed刷新文本请求成功率**提升0.29%**，Feed刷新图片请求成功率**提升0.23%**，也是非常不错的收益。

六、结语

连接优化是个持续性的话题，没有最优只有更优。上面介绍的百度App的一些经验和做法并不见得完美，但我们会继续深入的优化下去，持续提升百度App的网络性能。

以上优化由百度App团队，内核团队，OP团队共建完成。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《三》弱网优化，敬请期待。

七、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://tools.ietf.org/html/rfc7918 False Start

4. https://tools.ietf.org/html/rfc5077 Session Resumption

5. https://tools.ietf.org/html/rfc7413 TCP Fast Open

拓展阅读：

百度App网络深度优化系列《一》DNS优化