链滴

OkHttp 中文文档之 Connections(原创汉化搬运)

Although you provide only the URL, OkHttp plans its connection to your webserver using thr
e types: URL, Address, and Route.

尽管你只提供了一个URL,但是OkHttp计划使用三种方式连接到你的服务器:URL,Address以及Route.

URLs

URLs (like https://github.com/square/okhttp) are fundamental to HTTP and the Internet. In a
dition to being a universal, decentralized naming scheme for everything on the web, they also
specify how to access web resources.

URLs (比如 https://github.com/square/okhttp) 是组成internet和HTTP的基础.它除了是web上所
事物基础的,分散的命名规则外,还指定如何访问web资源.

URLs are abstract:

- They specify that the call may be plaintext (http) or encrypted (https), but not which crypt
graphic algorithms should be used. Nor do they specify how to verify the peer's certificates (t
e HostnameVerifier) or which certificates can be trusted (the SSLSocketFactory).
- They don't specify whether a specific proxy server should be used or how to authenticate wi
h that proxy server.

URLs是抽象的:

- 它们指定一次网络调用是明文方式(http)还是加密方式(https),但是却不指定哪种安全协议算法该
使用.也不指定如何验证对方的证书(HostnameVerifier),或者哪种证书可以被信任(SSLSocketFactory).
- 它们也不指定是否应该使用一个特定的代理服务器进行身份验证,或者如何使用代理服务器进行身份
证.

They're also concrete: each URL identifies a specific path (like /square/okhttp) and query (like
q=sharks&lang=en). Each webserver hosts many URLs.

它们也是具体的(**注:真他喵的拗口**):每个URL都确定了一个独特的路径(比如/square/okhttp)和查询参数
比如?q=sharks&lang=en).每个服务器都有多个URLs.

Addresses

Addresses specify a webserver (like github.com) and all of the **static** configuration necessary
o connect to that server: the port number, HTTPS settings, and preferred network protocols (l
ke HTTP/2 or SPDY).

Addresses指定了一个网络服务器(比如github.com)以及需要连接到这个服务器的所有静态配置:端口
,HTTPS配置,以及首选的网络协议(比如HTTP/2 或者 SPDY).

URLs that share the same address may also share the same underlying TCP socket connection.
Sharing a connection has substantial performance benefits: lower latency, higher throughput
due to TCP slow start) and conserved battery. OkHttp uses a ConnectionPool that automatical
y reuses HTTP/1.x connections and multiplexes HTTP/2 and SPDY connections.

共享相同地址的URLs也可以共享相同的底层TCP套接字连接。共享一个连接有巨大的性能优势:低延迟
高吞吐量(由于TCP慢启动)和守恒的电池(**注:守恒的电池是什么鬼?我猜是更小的资源开销?**).OkHttp使用
接池自动重用HTTP/1.x连接和多路传输的HTTP/2以及SPDY连接.

In OkHttp some fields of the address come from the URL (scheme, hostname, port) and the re t come from the OkHttpClient.

在OkHttp中,address中的一些字段来自于URL(scheme,hostname,port(**注:拿http://www.baidu.c m:80来举例:scheme为http,hostname为www.baidu.com,port为80,一般浏览器中http请求默认80端口,可省略,同理https默认请求端口为443,通常情况下也可省略**)),剩下的部分由OkHttpClient构建.

Routes

Routes supply the **dynamic** information necessary to actually connect to a webserver. This is he specific IP address to attempt (as discovered by a DNS query), the exact proxy server to us (if a ProxySelector is in use), and which version of TLS to negotiate (for HTTPS connections).

Routes提供真实连接到一个网络服务器所需的**动态**信息:用来尝试访问的明确的IP地址(通过一次DNS 询发现),使用准确的代理服务器(如果ProxySelector被使用),以及使用哪个版本的TLS协议(对于HTTPS 接来说).

There may be many routes for a single address. For example, a webserver that is hosted in mu tiple datacenters may yield multiple IP addresses in its DNS response.

一个地址或许拥有多个路由,举例来说,一个网络服务器拥有多个数据中心,或许会有多个IP地址对DNS 问进行响应.(**注:通俗来说,就是对一个域名的请求可能会被转发到多个真实IP,负载分流之类的巴拉巴拉**)

Connections

When you request a URL with OkHttp, here's what it does:

1. It uses the URL and configured OkHttpClient to create an **address**. This address specifies how we'll connect to the webserver.
2. It attempts to retrieve a connection with that address from the **connection pool**.
3. If it doesn't find a connection in the pool, it selects a **route** to attempt. This usually mean making a DNS request to get the server's IP addresses. It then selects a TLS version and proxy server if necessary.
4. If it's a new route, it connects by building either a direct socket connection, a TLS tunnel (fo HTTPS over an HTTP proxy), or a direct TLS connection. It does TLS handshakes as necessary.
5. It sends the HTTP request and reads the response.

当你使用OkHttp请求一个URL,将有这些事情需要执行:

1. 使 URL和配置好的OkHttpClient传建一个 **address**.这个address指定我们如何连接到网络服务器.
2. 尝试从 **连接池**中重新取回一个请求此address的连接
3. 如果第二步没有找到可用的请求,就会选择一个路由进行尝试,这通常意味着进行一次DNS请求来获 这个网络服务器的真实IP地址,然后如果有必要的话,选择一个TLS协议版本和代理服务器.
4. 如果是一个新的路线(**注:意味着从第三步过来的**),就通过构建一个直接的TLS连接或者在代理访问情况下是一个直接的套接字连接和一个TLS隧道来完成对服务器的连接.
5. 发送HTTP请求,读取响应.

If there's a problem with the connection, OkHttp will select another route and try again. This a lows OkHttp to recover when a subset of a server's addresses are unreachable. It's also useful when a pooled connection is stale or if the attempted TLS version is unsupported.

当连接发生问题时,OkHttp将会选择另一个路由进行重新访问.在一些情况下这将可以使OkHttp恢复常,比如服务器的部分节点不可用,连接池中的连接已过期,TLS协议版本不再被支持等.

Once the response has been received, the connection will be returned to the pool so it can be reused for a future request. Connections are evicted from the pool after a period of inactivity.

当响应被接收后,连接将会被返回至连接池中,以便被后续可能的请求重复使用.如果连接池中的连接在段时间内不被使用,则将会被移除.

译:洗澡狂魔,原文wiki地址:https://github.com/square/okhttp/wiki/Connections