OkHttp&Retrofit接入HTTPDNS SDK

以下代码片段摘自SDK使用Sample（HttpDnsSample目录），完整代码请参考使用Sample

OkHttp接入HTTPDNS SDK

得益于OkHttp的良好设计，我们可以直接向OkHttp注入DNS实现

OkHttpClient.Builder类提供了dns方法，开发者可以传入Dns接口的自定义实现，来替换默认的系统DNS实现

示例代码如下：

```
// DnsServiceWrapper.kt
object DnsServiceWrapper {

    private val EMPTY_ADDRESSES = arrayOf<InetAddress>()

    private val proxyHost by lazy { System.getProperty("http.proxyHost") }
    private val proxyPort by lazy { System.getProperty("http.proxyPort") }
    private val useHttpProxy by lazy {
        @Suppress("LocalVariableName")
        val _useHttpProxy = null != proxyHost && null != proxyPort
        DnsLog.d("useHttpProxy: %b", _useHttpProxy)
        _useHttpProxy
    }

    val useHttpDns = BuildConfig.USE_HTTP_DNS

    fun getAddrsByName(hostname: String): Array<out InetAddress> {
        // 客户端启用HTTP代理时，不使用HTTPDNS
        if (useHttpProxy || !useHttpDns) {
            // LocalDNS只取第一个IP
            return getAddrByNameByLocal(hostname)?.let { arrayOf(it) } ?:
EMPTY_ADDRESSES
        }
        val ipSet = DnsService.getAddrsByName(hostname, false)
        if (IpSet.EMPTY == ipSet) {
            return EMPTY_ADDRESSES
        }
```

```
        // 当前v6环境质量较差，优先选择v4 IP，且只考虑使用第一个v6 IP
        return when {
            ipSet.v6Ips.isNotEmpty() && ipSet.v4Ips.isNotEmpty() ->
                arrayOf(
                    *(ipSet.v4Ips.map { InetAddress.getByName(it)
}.toTypedArray()),
                    InetAddress.getByName(ipSet.v6Ips[0])
                )
            ipSet.v6Ips.isNotEmpty() ->
arrayOf(InetAddress.getByName(ipSet.v6Ips[0]))
            ipSet.v4Ips.isNotEmpty() -> ipSet.v4Ips.map {
InetAddress.getByName(it) }.toTypedArray()
            else -> EMPTY_ADDRESSES
        }
    }

    private fun getAddrByNameByLocal(hostname: String) =
        try {
            InetAddress.getByName(hostname)
        } catch (e: UnknownHostException) {
            null
        }
}

// OkHttpHelper.kt
internal object OkHttpHelper {

    private val dns by lazy {
        Dns { hostname ->
DnsServiceWrapper.getAddrsByName(hostname).toMutableList() }
    }

    val okHttpClient: OkHttpClient by lazy {
        OkHttpClient
            .Builder()
            .dns(dns)
            .build()
    }
}
```

Retrofit + OkHttp接入HTTPDNS SDK

Retrofit实际上是一个基于OkHttp，对接口做了一层封装桥接的库。因此只需要仿照OkHttp的接入方式，定制Retrofit中的OkHttpClient，即可方便地接入HTTPDNS SDK

示例代码如下：

```
// RetrofitHelper.kt
internal object RetrofitHelper {
```

```
    private val dns by lazy {
        Dns { hostname ->
DnsServiceWrapper.getAddrsByName(hostname).toMutableList() }
    }

    private val okHttpClient by lazy {
        OkHttpClient
            .Builder()
            .dns(dns)
            .build()
    }

    // ...

    val retrofit: Retrofit by lazy {
        Retrofit
            .Builder()
            .client(okHttpClient)
            // ...
            .build()
    }
}
```

注意事项

客户端如果网络库统一的话，则我们注入的DNS实现很可能会承载所有的客户端网络流量的域名解析工作

业务侧如果只是**部分**域名需要通过HTTPDNS进行域名解析保障的话，最好在SDK初始化时配置域名白名单，只有需要进行解析保障的域名才通过HTTPDNS SDK接口进行域名解析，避免不必要的流量使用

初始化HTTPDNS SDK时，可以调用

```
// 域名白名单支持通配符形式进行配置，如"*.qq.com"即以qq.com为后缀的域名都允许使用HTTPDNS服务进行域名解析
DnsConfig.Builder /* DnsConfig.Builder. */protectedDomains(String...
domains);
```

接口来配置域名白名单