[SCA] 网络诊断技术方案设计

文档历史

版本	修订日期	修订者	修订内容
1.0	2021.1.20	杨涛	初稿

1、需求概况

1.1 需求背景

由于移动端应用的功能越来越依赖网络状况，为了能随时反馈移动端网络状况，并且在用户环境出现网络问题时及时给出解决方案，需要加入移动端的网络诊断功能。

1.2 需求目的

获取用户移动端网络状况，收集用户网络检查结果，以便在用户环境出现网络问题时及时给出解决方案。

1.3 需求功能

经调研，网络诊断需要收集的信息可包含如下信息：本机设备的基本信息（Device），基本网络详细信息（Net），网络通畅性和稳定性（Ping），Http请求检查（Http），检查本机Host（Host），检查常见端口（Port Scan），检查所经路由情况（TraceRoute），检查所经路由的传输单元（Mtu Scan），检查DNS域名解析（NSLookup）。除去Device和Net信息，其他都是针对特定服务器（如WMS服务器的不同环境）进行检测。具体实现见详细方案。

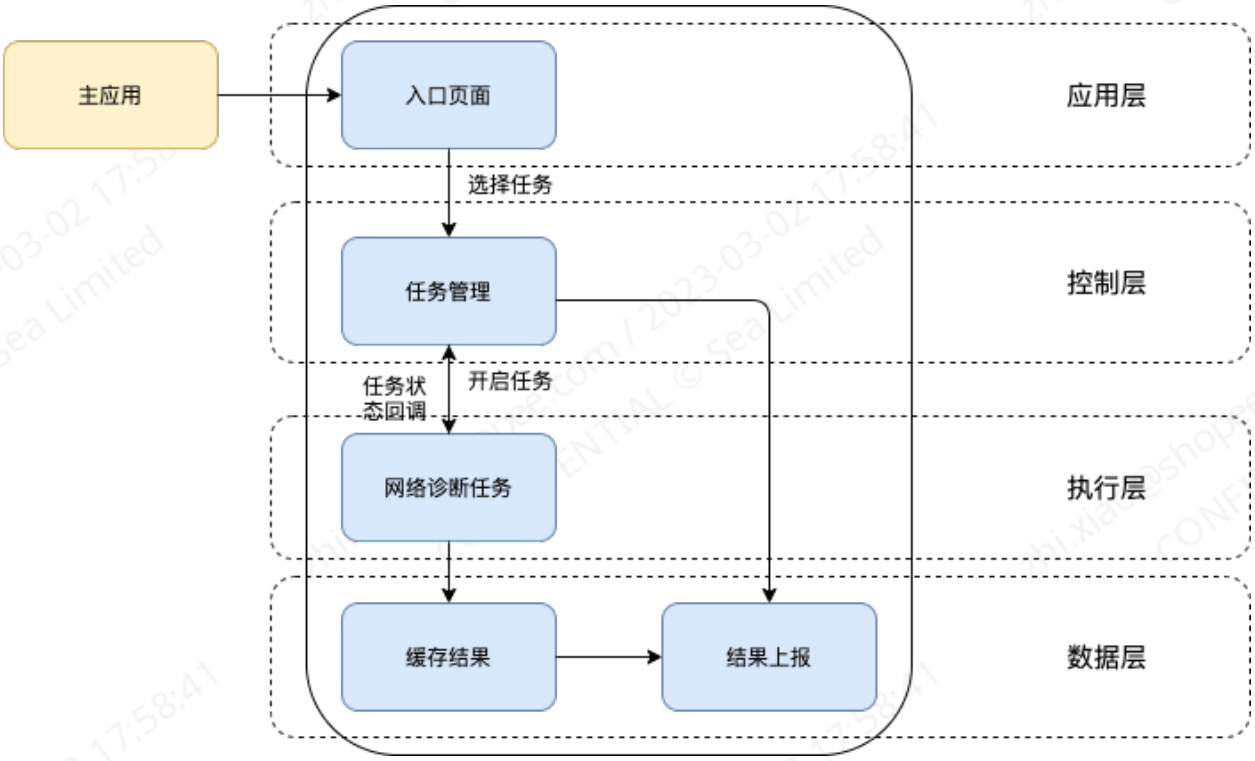
调研文档：https://docs.google.com/document/d/1InyQW7OaejfGxNYCqT0VKLh_-DOehVot4BAJ4Wlhb_k

2、技术方案设计

2.1 整体方案设计

网络诊断模块可以做成应用SDK，方便接入各个应用。

整体架构如下图

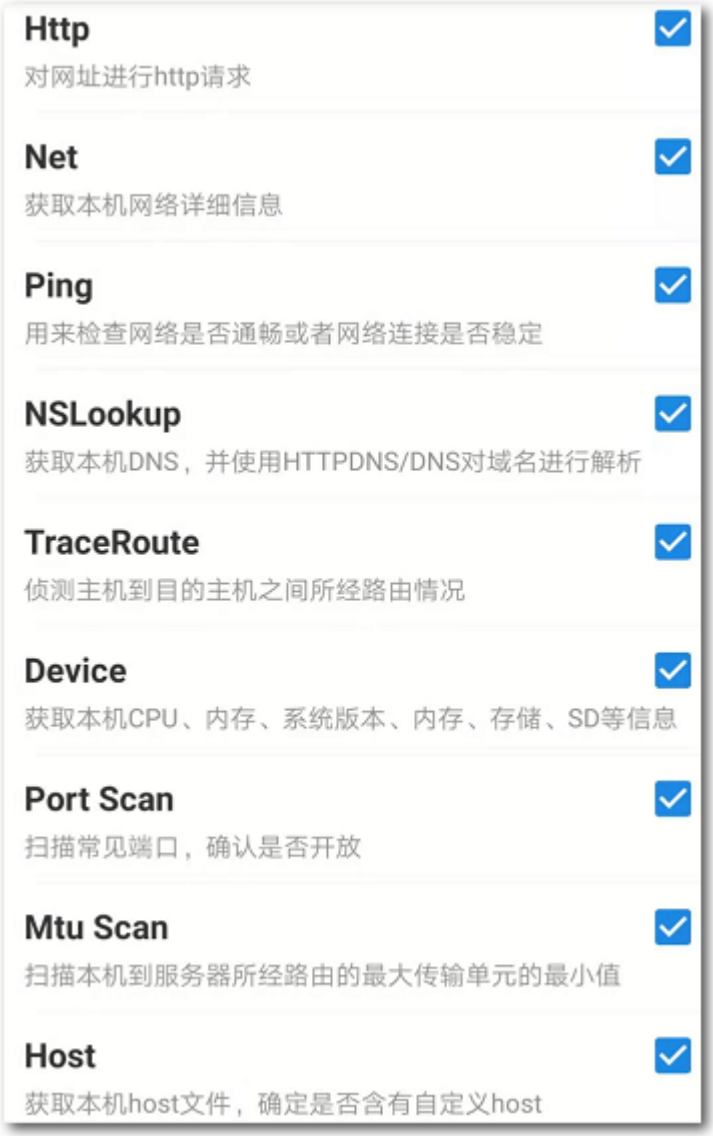


功能入口和触发：

1. 设置固定入口。

2. 如果接口缓慢，根据接口请求timeout时间（后台可配置），比如超过5秒，弹框提示是否进行网络诊断。

入口页面可参考腾讯云App界面，可选择诊断的条目：



控制层管理任务的执行，包含线程控制，任务状态管理等。执行层是具体执行每一项检查。检查结果则通过数据层先存到本地数据库，任务完成后再将结果全部上传，可上传至Android日志平台。

2.2 详细方案设计

主要是执行层的详细实现方案。

2.2.1 Device

设备基本信息包括：内存、CPU、存储、SD卡、系统版本等信息。

其中内存信息可以从系统的/proc/meminfo文件获取，包含内存大小、内存剩余、缓存大小等信息。

内部存储和SD卡信息，可以通过Environment类获取内部存储或外部存储的总目录文件，从而分别获得内部存储和SD卡的总空间和剩余空间等。

电池电量，可以通过广播Intent.ACTION_BATTERY_CHANGED获取，拿到的intent可以得到当前电量百分比intent.getIntExtra("level", 0)。

CPU信息可以从系统的/proc/cpuinfo文件获取，包含CPU的型号、频率等信息。

系统信息，其中版本信息可以从系统的/proc/version文件获取，包含设备型号、内核版本等信息。当前语言和语言语言列表可以从Locale类获取。厂商信息、Android版本号可以从android.os.Build类获取。IMEI信息则需要获取电话权限，从TelephonyManager的deviceID获取。

2.2.2 Net

网络基本信息包含：网络是否可用、网络类型、流量网络类型、WIFI信息、手机信号信息、本地IP、出口IP、出口IP归属、本地DNS、出口DNS、出口DNS归属等。

网络可用性，用ConnectivityManager获取NetworkInfo，判断状态是否连接。也可以从用ConnectivityManager中获取网络类型，是WIFI还是移动流量。

移动流量网络具体的类型，可以通过TelephonyManager获取，具体类型参考下表：（29及以上，需要电话权限，否则获取不到）

状态	值	类型
NETWORK_TYPE_UNKNOWN	0	不知名的网络
NETWORK_TYPE_GPRS	1	2G(2.5)
NETWORK_TYPE_EDGE	2	2G(2.75G)
NETWORK_TYPE_UMTS	3	3G WCDMA 联通
NETWORK_TYPE_CDMA	4	2G 电信
NETWORK_TYPE_EVDO_0	5	3G
NETWORK_TYPE_EVDO_A	6	3G的过渡（3.5G）
NETWORK_TYPE_1xRTT	7	2G的过渡
NETWORK_TYPE_HSDPA	8	3.5G
NETWORK_TYPE_HSUPA	9	3.5G
NETWORK_TYPE_HSPA	10	3G (分HSDPA,HSUPA)
NETWORK_TYPE_IDEN	11	2G
NETWORK_TYPE_EVDO_B	12	3G
NETWORK_TYPE_LTE	13	4G
NETWORK_TYPE_EHRPD	14	3G
NETWORK_TYPE_HSPAP	15	3G 比 HSDPA 快些
NETWORK_TYPE_GSM	16	2G 通用的移动联通电信2G模式
NETWORK_TYPE_TD_SCDMA	17	3G 移动
NETWORK_TYPE_IWLAN	18	4G
NETWORK_TYPE_LTE_CA	19	4G
NETWORK_TYPE_NR	20	5G

WIFI的信号强度和信号等级，可以通过WifiManager获取，manager.getConnectionInfo().getRssi()。得到的值是一个0到-100的区间值，是一个int型数据，其中0到-50表示信号最好，-50到-70表示信号偏差，小于-70表示最差，有可能连接不上或者掉线。

WifiManager还能获取如下信息：

wifiinfo.getSSID()；//获取WIFI名字。

wifiinfo.getIpAddress()；//获取IP地址。

wifiinfo.getMacAddress()；//获取MAC地址。

wifiinfo.getNetworkId()；//获取网络ID。

wifiinfo.getLinkSpeed()；//获取连接速度。

本地IP相关，可以通过NetworkInterface网络虚拟接口获得，包含IPv4和IPv6。

本地DNS，SDK版本大于等于21，可以用ConnectivityManager获取，获取活跃网络，通过活跃网络拿到连接属性，连接属性就能拿到DnsServers，从而得到DNS的host地址；如果不成功或者Android版本较低，则可以尝试从系统属性获取，通过反射拿到android.os.SystemProperties的类，DNS服务器对应的系统属性为，“net.dns1”、“net.dns2”、“net.dns3”、“net.dns4”，再通过InetAddress.getByName(server)拿到具体地址；如果还不成功，则可以通过控制台指令获取，运行控制台指令：Runtime.getRuntime().exec("getprop")，读取输出流，从属性中过滤“.dns”结尾的字符串，可以得到DNS服务器，再通过InetAddress.getByName(server)拿到具体地址。

出口IP和DNS及其归属地，可以通过访问专门的网络获取，可以参考使用网易的工具：https://nstool.netease.com/，访问后可以获得一个专有网址，可以从html信息中解析获得，示例返回：（下划线已标出专有网址）

<html><head><title>网易DNS检测工具</title></head><body><iframe src='https://only-373718-58-250-178-135.nstool.zhuanzfx.com/' frameborder=0 scrolling=no height='100%' width='100%'></iframe></body></html>

访问专有网址可以得到具体的出口IP和DNS，及其归属地。

移动信号相关，需要定位权限，可以从TelephonyManager获取，getAllCellInfo()可以获得所有基站相关信息，根据设备支持的网络制式，一般会有GSM、CDMA、LTE、WCDMA，分别对应CellInfoGsm、CellInfoCdma、CellInfoLte和CellInfoWcdma，getDbm()获取信号强度，getLevel()获取信号等级，getAsuLevel()获取信号电平值等。

2.2.3 Ping

可以利用控制台指令，运行/system/bin/ping的Ping程序，以实现Ping。除了指明特定服务器的Host，还可以指定Ping的次数（发送包数量），以及超时时间。建议次数大于十次，超时时间半分钟到一分钟。

从输出流中解析结果，可以得到目标服务器的IP地址，发送包、接收包、丢包率、TTL生存时间、最大RTT、最小RTT以及平均RTT等。

示例输出：

PING www.a.shifen.com (14.215.177.39) 56(84) bytes of data.

64 bytes from 14.215.177.39: icmp_seq=1 ttl=53 time=8.23 ms

64 bytes from 14.215.177.39: icmp_seq=2 ttl=53 time=27.4 ms

64 bytes from 14.215.177.39: icmp_seq=3 ttl=53 time=24.8 ms

64 bytes from 14.215.177.39: icmp_seq=4 ttl=53 time=23.5 ms

64 bytes from 14.215.177.39: icmp_seq=5 ttl=53 time=22.9 ms

64 bytes from 14.215.177.39: icmp_seq=6 ttl=53 time=27.7 ms

64 bytes from 14.215.177.39: icmp_seq=7 ttl=53 time=25.5 ms

64 bytes from 14.215.177.39: icmp_seq=8 ttl=53 time=25.7 ms

64 bytes from 14.215.177.39: icmp_seq=9 ttl=53 time=30.5 ms

64 bytes from 14.215.177.39: icmp_seq=10 ttl=53 time=27.7 ms

--- www.a.shifen.com ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9016ms

rtt min/avg/max/mdev = 8.231/24.451/30.539/5.806 ms

2.2.4 Http

主要检查对目标服务器的Http访问，获得访问结果（状态码，返回头部，用时等），还可以计算网速信息。

Http可以使用底层HttpURLConnection的实现，若考虑使用应用当前的网络框架，侵入性会比较大，需要考虑如何跟网络诊断模块交互。

网速计算，可以通过流量相关接口提供的数据来计算。经过已知的时间段，流量差除以时间及当前网速。流量数据参考TrafficStats类获取：

```
/** 获取手机通过 2G/3G 接收的字节流量总数 */

TrafficStats.getMobileRxBytes();

/** 获取手机通过 2G/3G 接收的数据包总数 */

TrafficStats.getMobileRxPackets();

/** 获取手机通过 2G/3G 发出的字节流量总数 */
```

```
TrafficStats.getMobileTxBytes();

/** 获取手机通过 2G/3G 发出的数据包总数 */

TrafficStats.getMobileTxPackets();

/** 获取手机通过所有网络方式接收的字节流量总数(包括 wifi) */

TrafficStats.getTotalRxBytes();

/** 获取手机通过所有网络方式接收的数据包总数(包括 wifi) */

TrafficStats.getTotalRxPackets();

/** 获取手机通过所有网络方式发送的字节流量总数(包括 wifi) */

TrafficStats.getTotalTxBytes();

/** 获取手机通过所有网络方式发送的数据包总数(包括 wifi) */

TrafficStats.getTotalTxPackets();

/** 获取手机指定 UID 对应的应程序用通过所有网络方式接收的字节流量总数(包括 wifi) */

TrafficStats.getUidRxBytes(uid);

/** 获取手机指定 UID 对应的应用程序通过所有网络方式发送的字节流量总数(包括 wifi) */

TrafficStats.getUidTxBytes(uid);
```

2.2.5 Host

主要检查localHost，包括IPv4和IPv6，直接读取系统文件/system/etc/hosts即可获取。

2.2.6 Port Scan

端口扫描，可以用UDP或TCP的方式，扫描端口可以自定义一个范围，也可以全端口1-65536。

由于扫描的端口较多，可以用线程池并行扫描，选用FixedThreadPool，核心线程数可以设置大一些，awaitTermination来等待所有端口扫描完成，时间可以根据要扫描的端口数来设定。

UDP的方式，可以使用DatagramPacket和DatagramSocket来实现，通过和DatagramSocket与目标服务器建立连接，发送和接收DatagramPacket包，可获取总的耗时，是否成功的信息。

TCP的方式，直接使用Socket，连接目标服务器即可——socket.connect(new InetSocketAddress(inetAddress, portNo), timeoutMillis)。同样可获取总的耗时，是否成功的信息。

2.2.7 TraceRoute

Trace Route主要利用增加存活时间（TTL）值来实现，每当数据包经过一个路由器，其存活时间就会减1。当其存活时间是0时，主机便取消数据包，并传送一个ICMP TTL数据包给原数据包的发出者。

用Ping指令可以比较简单地实现上述过程。Ping指令先发送探测包，指定TTL从1开始，每次加一，依次检查所有路由，指令参考“/system/bin/ping -c 1 -w 1 -t [TTL] [Host]”。若未到达目的地，返回一个Time to live exceeded的返回，示例如下：

PING www.a.shifen.com (14.215.177.39) 56(84) bytes of data.From 192.168.7.254: icmp_seq=1 Time to live exceeded--- www.a.shifen.com ping statistics ---1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

此时可以得到当前到达的路由IP地址，可以再用Ping指令检查主机跟此路由的连通，并记录RTT。不断增加TTL直到到达目的服务器。到达目的服务器的返回是正常的Ping返回，可以参考正则判断：包含“(?<=from).*(?=: icmp_seq=1 ttl=)”则为正常返回。另外，TTL的增加也可以设置一个循环上限，防止由于网络问题导致的无限循环，通常设置30以上均可。

2.2.8 Mtu Scan

Maximum Transmission Unit，中文名是：最大传输单元。MTU限制的是数据链路层的payload，也就是上层协议的大小，例如IP，ICMP等。MTU扫描是扫描与目标服务器之间所有的路由，其最大传输单元的最小值。在Ethernet中，MTU为1500字节；在FDDI中，MTU为4352字节；在 IP over ATM中，MTU为9180字节。我们可以考虑1500这个标准，这是考虑传输时间和传输效率的一个折中值。

检测的策略，可以用Ping来检测，Ping可以给目标服务器发送特定大小的包，能发成功则表示MTU大于等于该包大小。可以从1500开始，如果发送失败则减小xBytes重新发送，直到成功或者包小于等于100字节，成功时的值就是最大传输单元的最小值。x的粒度可以自定义，越小则越精准，但越大运行时间就可能越长。

Ping指令可以参考：/system/bin/ping -c 1 -w 1 -s [mtu size] [host]，发送一个包，超时时间为1秒。具体执行可以参考第四部分。解析结果时，只要包含“df”字符串（注意忽略大小写）就说明发送失败需要拆包，其他情况则是成功。

2.2.9 NSLookup

主要检查DNS域名解析，用不同解析策略对目标服务器的域名进行解析。

主要的策略包含两种，一种是默认策略，一种是指定本地DNS地址的方式。

其中默认方式是，直接利用系统的InetAddress.getAllByName(host)方法获取IP地址，其中会包含IPv4和IPv6的两种IP地址。

指定DNS服务器的方式，通常是指定本地获取的DNS服务器地址（参考第三部分）。具体实现可以参考dnsjava: https://github.com/dnsjava/dnsjava。使用起来很简单：

```
Lookup lookup = new Lookup([目标服务器Host]);

SimpleResolver simpleResolver = new SimpleResolver([指定DNS服务器IP]);

lookup.setResolver(simpleResolver);

lookup.run();

Record[] records = lookup.getAnswers(); // 获取处理结果
```

结果包含处理得到的IP，TTL，域名等。

2.3 数据存储设计

数据库实现可以用比较成熟的GreenDao框架。

具体的表设计，一个表中记录诊断类型和结果字符串，以及诊断时间、Host等。上报结果按时间纬度，上报最新的检查结果，上传成功后可删除本地数据库数据。

定时检测，若网络可用时，拉取本地所有未上报成功的数据进行批量上报，上报成功则删除本地所有数据；若本地无数据则不进行上报。默认周期为一天，该时间可配置。