

如何做 Android 应用流量测试

前言

我们经常手机应用有这样的困惑：想知道应用费不费流量；想知道某几款同类应用，做同样的事儿，哪个更省流量；更深入的，想知道一款应用为什么这么费流量，流量都消耗在哪了；想知道在大 4G 时代，一觉醒来怎么房子车子就变成别人的了。。本文将介绍给您，解答上述困惑的简单方法。

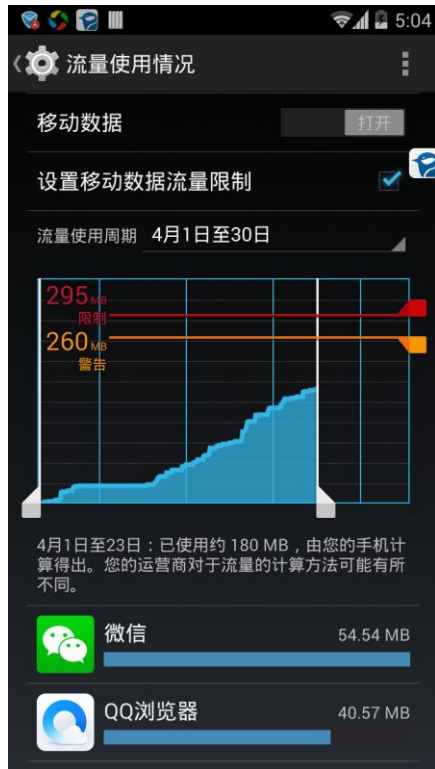
工具

GT (中文产品名称: 随身调)：是腾讯出品的开源调试工具，本次测试中用其进行手机的流量统计和抓包。请在 Android 手机上安装 GT 应用（可以通过官网或应用宝下载）。

Wireshark：抓包的分析工具，也提供了 Android 手机的抓包实现，GT 中抓包的功能就是在其提供的实现基础上的易用性封装，本次测试中用 Wireshark 进行抓包的分析。请在 PC 上安装 Wireshark。

正文

其实想知道一款应用费不费流量，大部分 Android4.x 版本系统已经可以简单的查看了：

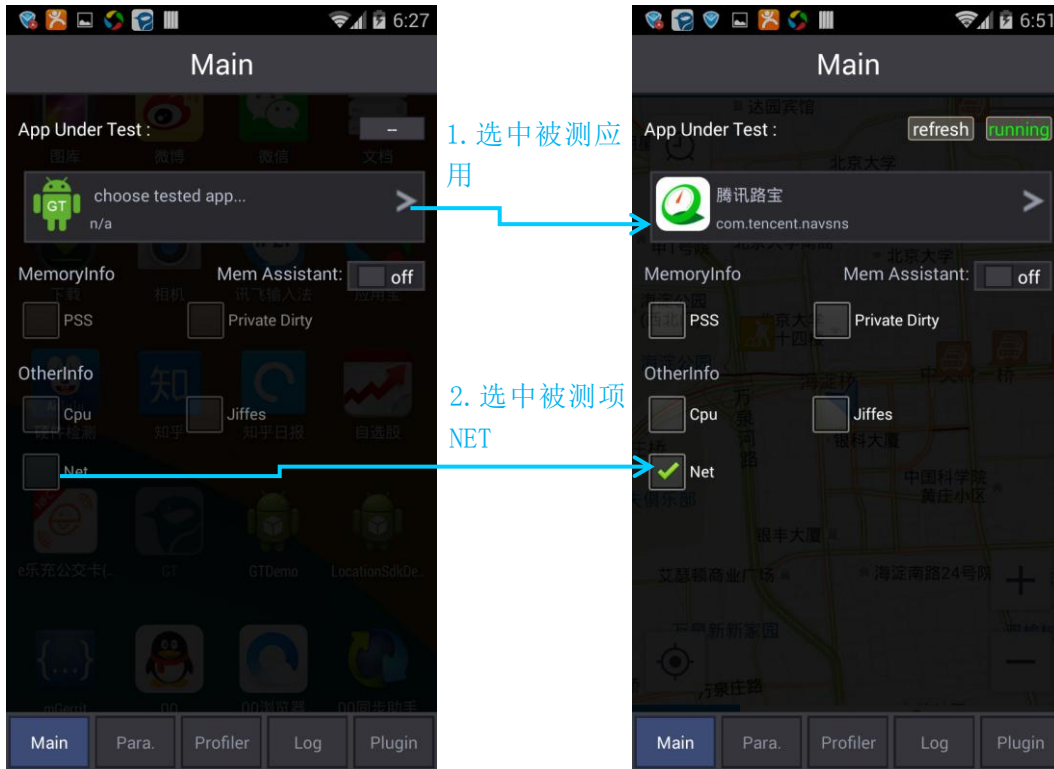


关注流量比较粗的话，看一下上面这里也就够了，但从测试的需求看，这里只能观察到宏观的流量情况，到 1 天的流量消耗就没法再细化了，如果想知道具体一个业务操作或一段时间内的流量消耗呢？如果想知道应用一次启动的流量消耗呢？这时就该使用前面介绍的工具了。

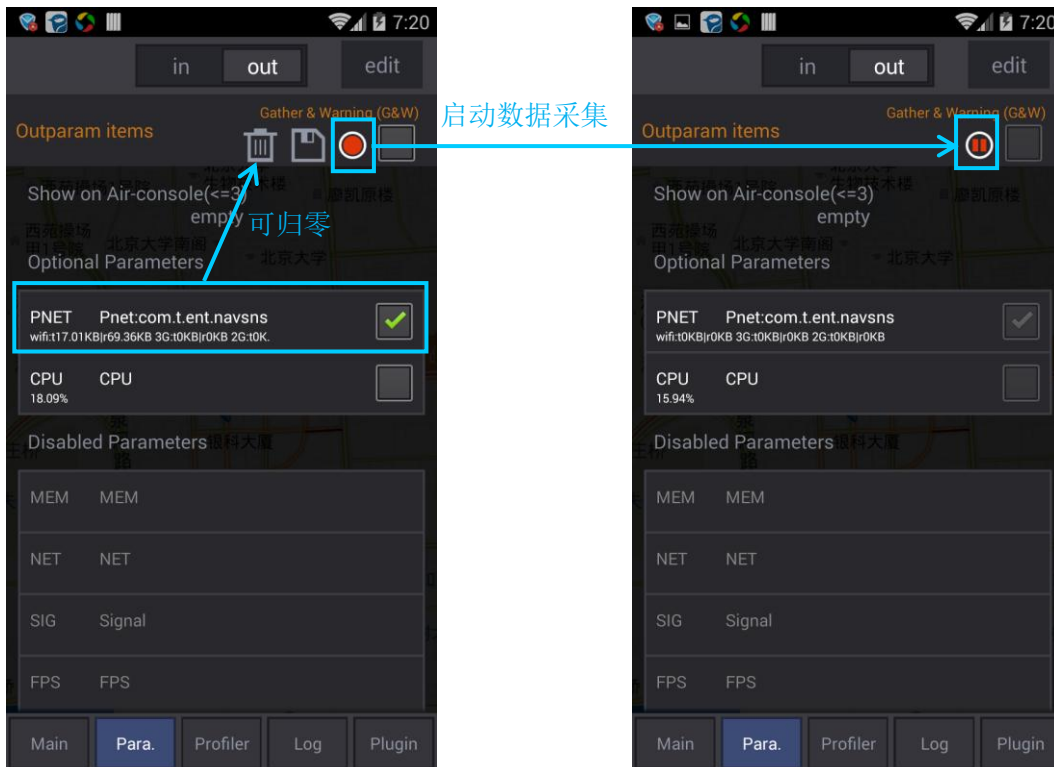
比前面稍微深入一些，我们可能需要知道一个业务操作过程内，消耗的流量，及发出请求的流量、收到响应结果的流量各有多少，并且流量的消耗曲线是怎样一个走势。这时就该使用 GT，关于 GT 的基本使用和为什么用 GT，[GT 网站](#)有详细的说明，这里只介绍和流量相关的部分。GT 提供了一种简单的测试方式，也提供了一个严谨但麻烦的测试方式。

- **首先我们来看简单的方式：**

1. 先将应用运行起来，然后启动 GT 并在 GT 上选中被测应用及被测项 NET（流量）。

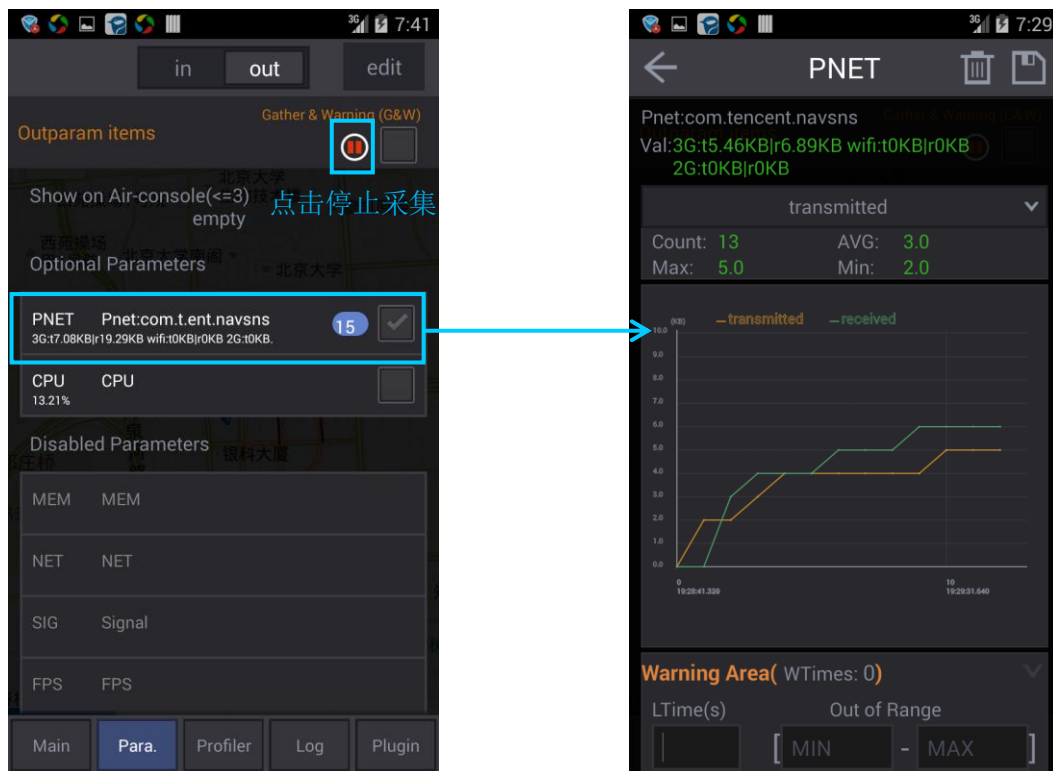


- 业务操作前，启动数据采集，将会记录选中应用的流量的变化，为了方便统计，可以先把业务操作前发生的流量记录归零。



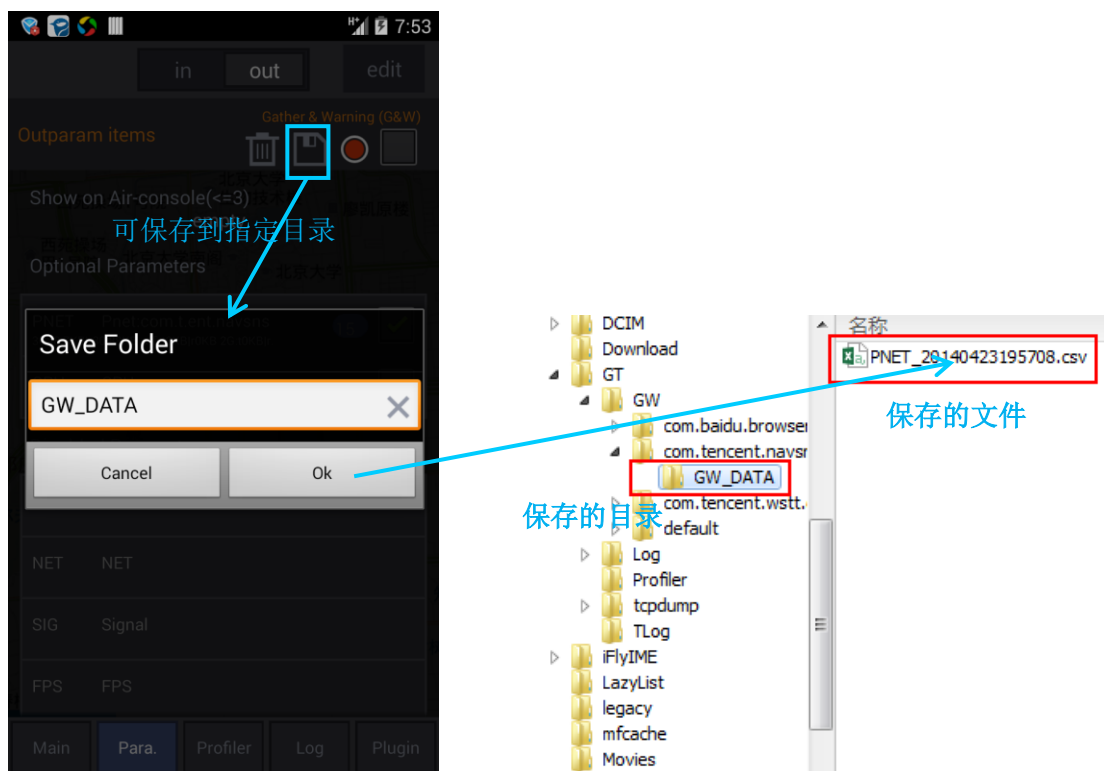
- 退到应用界面，执行需测试的业务操作。

4. 业务操作后，回到 GT 界面，停止流量数据的采集，查看本次业务操作流量的变化。



到这里，从前面一张图我们已经可以知道一个业务操作过程中消耗的流量，包括发出请求的流量、收到响应结果的流量、流量消耗曲线是怎样一个走势了。

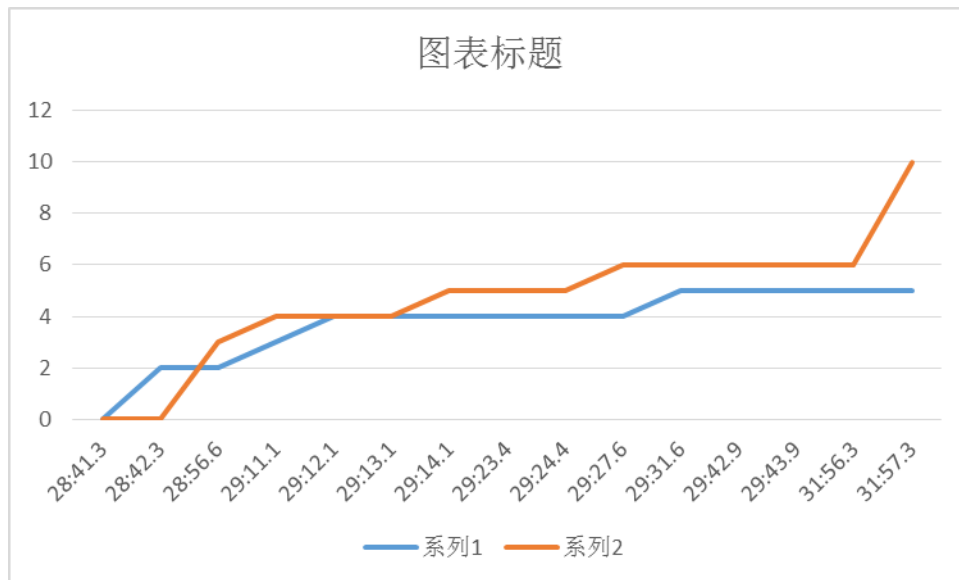
5. 我们可以保存本次测试结果到文件，以备后面更深入的分析。



参观一下这个文件：

	A	B	C	D
1	key	Pnet:com.tencent.navsns		
2	alias	PNET		
3	unit	(KB)		
4	begin date	#####		
5	end date	#####		
6	count	15		
7				
8		transmitt	received	
9	min	2	3	
10	max	5	10	
11	avg	3	4	
12				
13	28:41.3	0	0	
14	28:42.3	2	0	
15	28:56.6	2	3	
16	29:11.1	3	4	
17	29:12.1	4	4	
18	29:13.1	4	4	
19	29:14.1	4	5	
20	29:23.4	4	5	
21	29:24.4	4	5	
22	29:27.6	4	6	
23	29:31.6	5	6	
24	29:42.9	5	6	
25	29:43.9	5	6	
26	31:56.3	5	6	
27	31:57.3	5	10	
28				

用强大的 Excel 把 GT 应用里显示的趋势图还原出来不是难事。



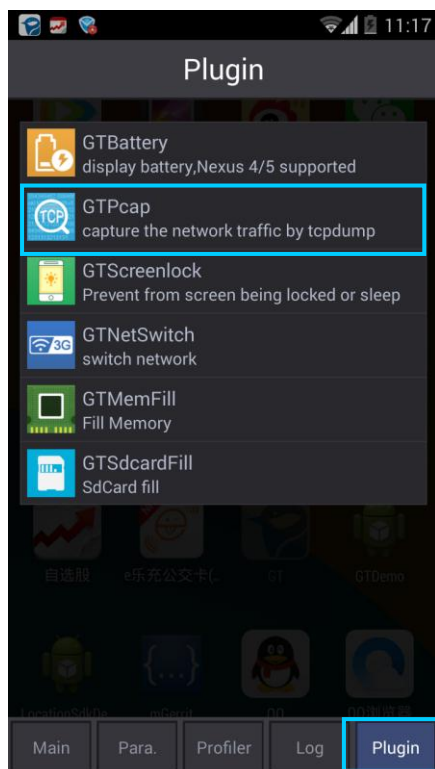
● 我们再来看看麻烦而严谨的方式：

如果只是纯粹测流量，上面的方式也足够了，那我们为什么需要麻烦而严谨的方式呢？这里有两个原因，一个是仅仅知道流量的大小和趋势，还不足以对后续流量优化进行明确的指导，即知道流量可能有点多，但不知道该如何着手优化。另一个原因是弥补上面方式的一个不足：有的应用，使用了本地 socket 和手机里其他进程产生交互，有时候

Android 系统会把这种手机内部的 socket 传输的数据量也计算到应用消耗的流量里(比如常见的视频应用不少都有这个问题), 此时上面的方式就显得不够准确了, 要获得真是网卡上发生的流量, 就需要抓包这种终极方法了。**注意掌握这种方法的前提是您得先掌握基础的 TCP 和 HTTP 网络知识。**

手机抓包是针对手机的网卡, 所以这种方式无法单独抓一个应用的包, 需要后续将归属于应用的包分析出来, 而为了后续分析减少工作量, 测试时候应尽量把其他能消耗流量的应用都关了。Android 手机的抓包是 Wireshark 提供的实现, GT 上面做了封装, 使手机可以不必连着 PC 即可抓包, 方便在室外测试的场景。

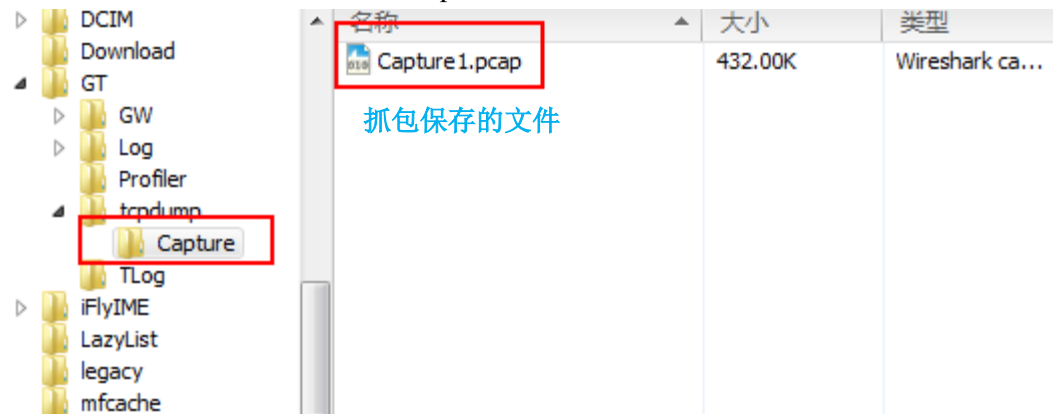
1. 先从 GT 启动抓包。



抓包功能的入口在这里

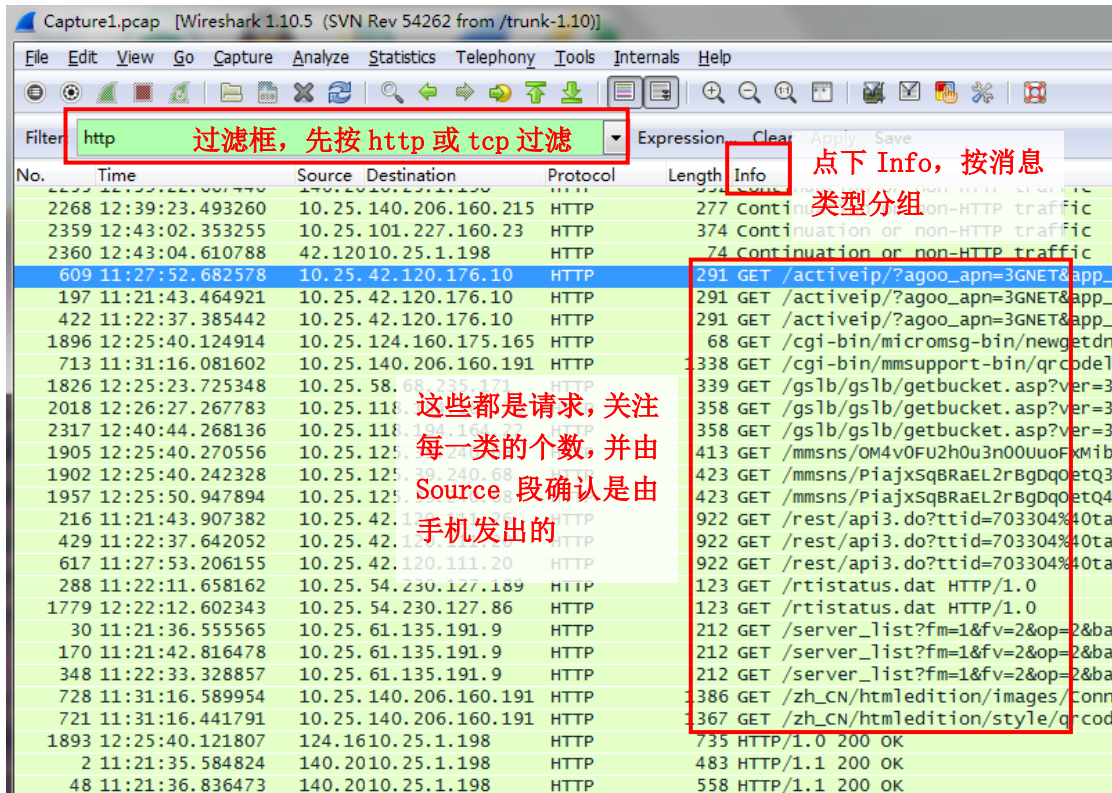


2. 之后还是执行测试的业务操作。
3. 被测业务操作结束后, 点击 stop, 即停止抓包, 并把抓包文件保存在对应的目录中。

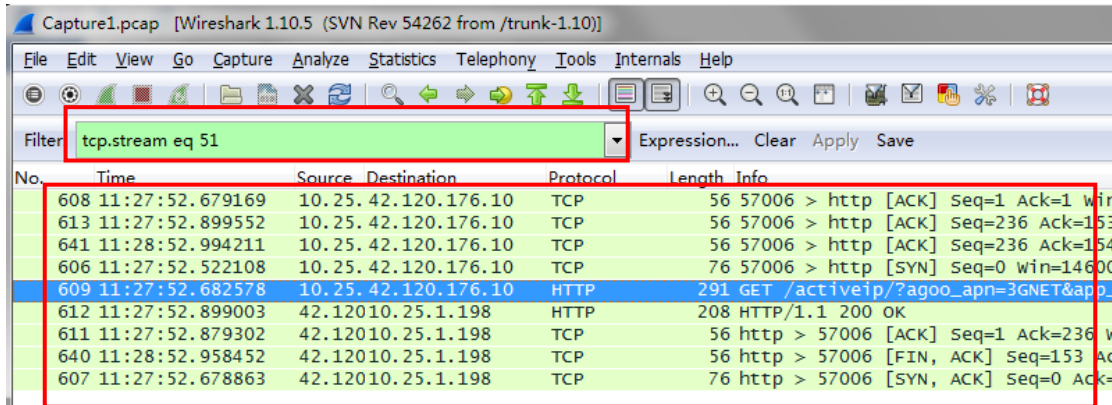


将抓到包文件导入到 PC, 用 Wireshark 即可分析抓包文件。关于 Wireshark 的使用, 和 PC 上的使用没有区别, 请大家自行在网上搜索, 这里仅对使用 Wireshark 的要点提示下:

- 1 我们最先需要知道我们的应用发出了哪些请求, 对应了上行流量, 可以在 Wireshark 左上角【过滤】框输入"http"或"tcp" (如果确认过被测应用都是 http 请求, 就只需要按 http 过滤), 确认测试场景 GET 和 POST 的请求**类型和个数** (过滤结果可按【Info】分类更方便统计)。



- 在具体请求上可以右键“follow tcp stream”, 等同于过滤条件 tcp.stream eq xx, 这样可以过滤出和它在同一个 TCP 流的消息。



- 过滤条件出来后再点击【统计】->【概要】, 对应 Bytes 栏【显示】列的数据即为流量。

Capture1.pcap [Wireshark 1.10.5 (SVN Rev 54262 from /trunk-1.10)]

File Edit View Go Capture Analyze **Statistics** Telephony Tools Internals Help

Summary

- Comments Summary
- Show address resolution
- Protocol Hierarchy
- Conversations
- Endpoints
- Packet Lengths...
- IO Graph
- Conversation List
- Endpoint List
- Service Response Time
- ANCP

Filter: tcp.stream eq 51

No.	Time	Source	Length	Info
608	11:27:52.679169	10.25	56	57006 >
613	11:27:52.899552	10.25	56	57006 >
641	11:28:52.994211	10.25	56	57006 >
606	11:27:52.522108	10.25	76	57006 >
609	11:27:52.682578	10.25	291	GET /act
612	11:27:52.899003	42.120	208	HTTP/1.1
611	11:27:52.879302	42.120	56	http > 5
640	11:28:52.958452	42.120	56	http > 5
607	11:27:52.678863	42.120	76	http > 5



- 通过对包的过滤分析，我们自然就可以得到流量的大小，产生流量的类型和原因，请求的频率，这样就能够对后续的流量优化进行指导了。

3C-环境限定1- 环境限定2	流量总消耗 (bytes)	类型	命令字	请求次数	流量消耗 (bytes)	过滤详情	备注
某应用-场景1	50394	GET	qt=rtt	7	30095	tcp.stream eq 1 or tcp.stream eq 18 or tcp.stream eq 27 or tcp.stream eq 33 or	3分钟一个
		GET	navtemp	4	18777	tcp.stream eq 0 or tcp.stream eq 51 or tcp.stream eq 47 or	15: 27~15: 28重复 三次才得到成功结果
		POST	wup				有连续1s俩的同一消息
		POST	lbsi?c=1& mars=1			http.request.uri contains "lbsi"	[TCP Retransmission] POST /lbsi?c=1&mars=1 HTTP/1.1 (application/x- www-form- urlencoded) 带着的 指数回退重传的, 我 们不需要管, 和被重 传消息是有关系的。 有重复消息
		POST	analytics /upload	1	1522		数据上报, 很小
竞品应用-场景1	0						

5 更谨慎的, 抓包和 GT 采集流量数据可以相互对照, 避免分析时有所遗漏。

The screenshot shows a Wireshark interface with a packet list on the left and a packet details pane on the right. The packet list contains various HTTP requests and responses. Red lines are drawn from the packet list to a table on the right, which appears to be a summary or index of the captured traffic. The table has columns for time, source, destination, protocol, length, and info, and is overlaid on a spreadsheet-like grid.

- **如何判断一个应用的流量消耗偏高**

如果看流量的绝对值看不出高低，那就找几个同类型的产品对比一下。如果完成同样的事务，被测应用比同类产品高很多，那就是偏高了，可能有优化空间。

- **如何找到有效的优化点**

把分析的不同类数据包，按包占总流量大小的比例，和包的数量排序，占比多的，和消息数量多的，一个优化空间大，一个精简请求次数的机会大。

- **常见的流量问题**

最后简单例举几类可控的比较容易优化的流量问题给大家：

- ◆ **冗余内容**

同类请求被间隔执行，请求的内容包含一些相对静态的信息，**正确的处理是第一次请求包括静态信息就好，后面的同类请求只包含必要的即时变化信息即可。错误的处理方式是每次请求服务器都返回一次静态信息。**

- ◆ **冗余请求**

有的时候会发现应用短时间内发出多个同样的请求，收到结果也都几乎一样，这种情况应该**尽量减少请求次数，同时注意排查程序逻辑错误**，也许问题不像表面看起来那么简单。

- ◆ **无用请求**

有的请求，你会发现谁也不知道它是干嘛的，**很可能是以前版本遗留下来的无用请求，或者是引用的其他代码包偷偷发出的，甚至是间谍请求**，请收集一切证据后，毫不犹豫的干掉它。

- ◆ **永远无法得到回应的请求**

如果见到某类请求**永远的连接失败或被返回 404 之类的失败结果**，那它不是历史遗留的多余请求，就是某个不易察觉的功能已经失效了。

- ◆ **过多的失败请求**

有见过一类或一组请求，n 个成功之中夹着 m 个失败的吗？举个简单的场景，某类请求，间隔 1 分钟后连续发两次，**总是先有一次失败的请求，1s 后马上再次发出一次同样的请求就成功了**（这里 1s 后发出的请求是指业务逻辑层判断前面请求失败后延迟 1s 后重传的）。很好奇为什么第一次总失败是吧，就有这么种情况，**客户端两次请求乐观的想要复用同一个 TCP 连接(长连接半长连接)，但是服务端不这么想，也许是客户端发起两次请求的间隔，超出了服务端设置的长连接无响应时限。**如何判断呢？看看失败的那次请求，是否和前一次成功的请求复用了同一个 TCP 连接（体现在 Wireshark 的 streamId）。

- ◆ **非预期请求**

比如一种常见的情况，应用退后台后，有些请求就没必要了，**观察下自己的产品，是否在后台真的没有发出这些请求。**