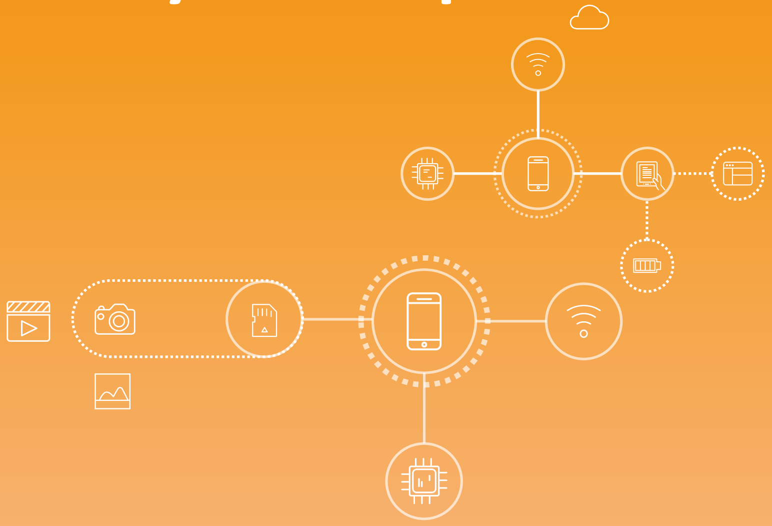
MEDIATEK

Network issue analysis sop



MEDIATEK

网络分析工具

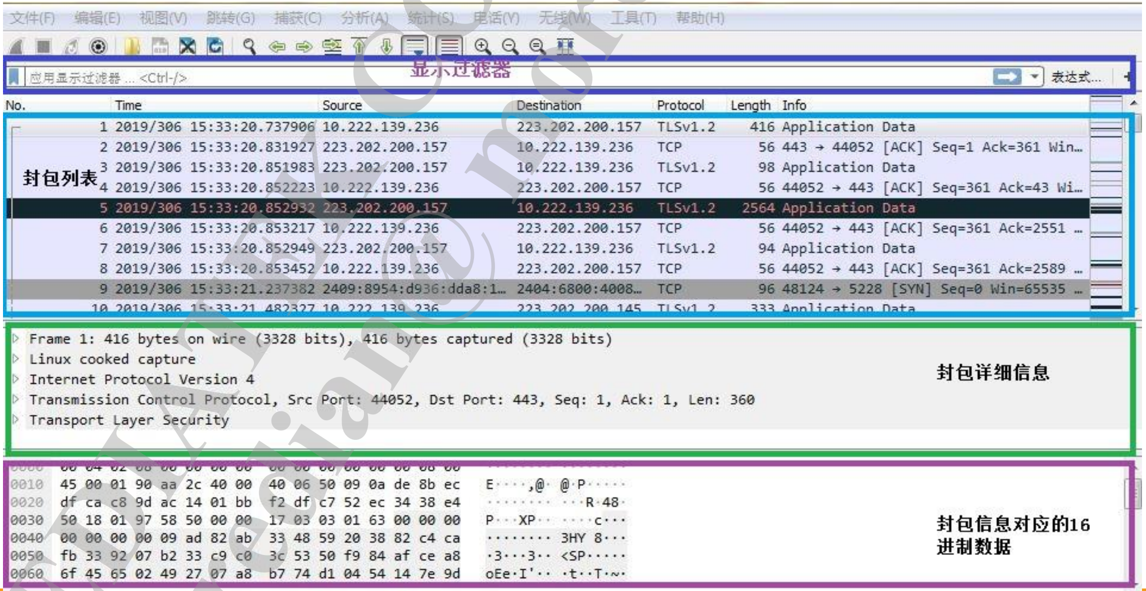
工欲善其事 必先利其器

网络分析工具

- wireshark

要想做好网络分析，必须先熟练掌握一种网络分析工具，当前最常用的网络分析工具是wireshark 。

wireshark下载路径：https://www.wireshark.org/download.html



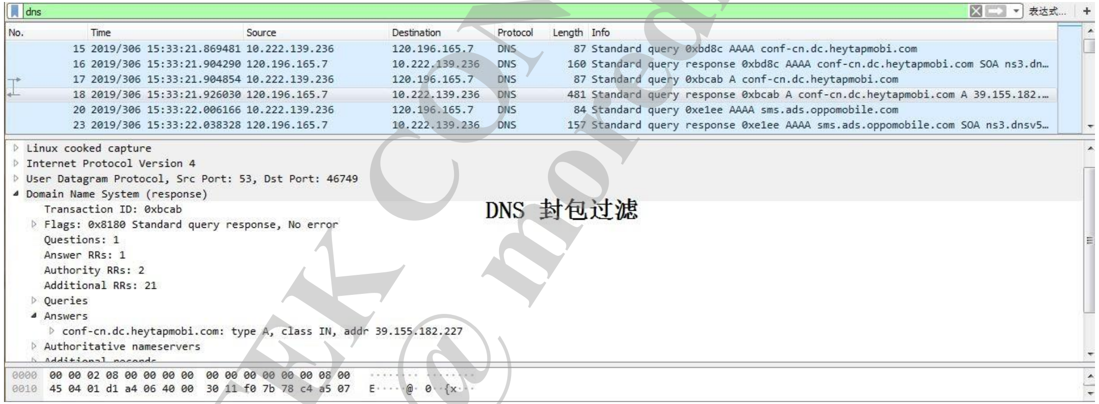
```
No.   Time                        Source                  Destination         Protocol  Length  Info
    1 2019/306 15:33:20.737906 10.222.139.236          223.202.200.157     TLSv1.2    416 Application Data
    2 2019/306 15:33:20.831927 223.202.200.157         10.222.139.236      TCP         56 443 → 44052 [ACK] Seq=1 Ack=361 Win…
    3 2019/306 15:33:20.851983 223.202.200.157         10.222.139.236      TLSv1.2     98 Application Data
    4 2019/306 15:33:20.852223 10.222.139.236          223.202.200.157     TCP         56 44052 → 443 [ACK] Seq=361 Ack=43 Wi…
    5 2019/306 15:33:20.852932 223.202.200.157         10.222.139.236      TLSv1.2   2564 Application Data
    6 2019/306 15:33:20.853217 10.222.139.236          223.202.200.157     TCP         56 44052 → 443 [ACK] Seq=361 Ack=2551 …
    7 2019/306 15:33:20.852949 223.202.200.157         10.222.139.236      TLSv1.2     94 Application Data
    8 2019/306 15:33:20.853452 10.222.139.236          223.202.200.157     TCP         56 44052 → 443 [ACK] Seq=361 Ack=2589 …
    9 2019/306 15:33:21.237382 2409:8954:d936:dda8:1… 2404:6800:4008…     TCP         96 48124 → 5228 [SYN] Seq=0 Win=65535 …
   10 2019/306 15:33:21.482327 10.222.139.236          223.202.200.145     TLSv1.2    333 Application Data
```

```
▷ Frame 1: 416 bytes on wire (3328 bits), 416 bytes captured (3328 bits)
▷ Linux cooked capture
▷ Internet Protocol Version 4
▷ Transmission Control Protocol, Src Port: 44052, Dst Port: 443, Seq: 1, Ack: 1, Len: 360
▷ Transport Layer Security
```

```
0010  45 00 01 90 aa 2c 40 00  40 06 50 09 0a de 8b ec   E····,@· @·P·····
0020  df ca c8 9d ac 14 01 bb  f2 df c7 52 ec 34 38 e4   ········ ···R·48·
0030  50 18 01 97 58 50 00 00  17 03 03 01 63 00 00 00   P···XP·· ····c···
0040  00 00 00 00 09 ad 82 ab  33 48 59 20 38 82 c4 ca   ········ 3HY 8···
0050  fb 33 92 07 b2 33 c9 c0  3c 53 50 f9 84 af ce a8   ·3···3·· <SP·····
0060  6f 45 65 02 49 27 07 a8  b7 74 d1 04 54 14 7e 9d   oEe·I'·· ·t··T·~·
```

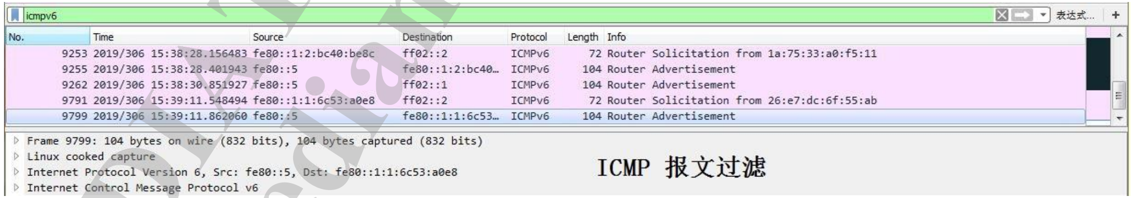
过滤器使用

使用过滤是非常重要的，初学者使用wireshark时，将会得到大量的冗余信息，在几千甚至几万条记录中，以至于很难找到自己需要的部分。通过在显示过滤器中输入正确的过滤条件会帮助我们在大量的数据中迅速找到我们需要的信息



```
No.        Time                       Source          Destination     Protocol  Length Info
        15 2019/306 15:33:21.869481 10.222.139.236  120.196.165.7   DNS          87 Standard query 0xbd8c AAAA conf-cn.dc.heytapmobi.com
        16 2019/306 15:33:21.904290 120.196.165.7   10.222.139.236  DNS         160 Standard query response 0xbd8c AAAA conf-cn.dc.heytapmobi.com SOA ns3.dn…
        17 2019/306 15:33:21.904854 10.222.139.236  120.196.165.7   DNS          87 Standard query 0xbcab A conf-cn.dc.heytapmobi.com
        18 2019/306 15:33:21.926030 120.196.165.7   10.222.139.236  DNS         481 Standard query response 0xbcab A conf-cn.dc.heytapmobi.com A 39.155.182.…
        20 2019/306 15:33:22.006166 10.222.139.236  120.196.165.7   DNS          84 Standard query 0xe1ee AAAA sms.ads.oppomobile.com
        23 2019/306 15:33:22.038328 120.196.165.7   10.222.139.236  DNS         157 Standard query response 0xe1ee AAAA sms.ads.oppomobile.com SOA ns3.dnsv5…

▷ Linux cooked capture
▷ Internet Protocol Version 4
▷ User Datagram Protocol, Src Port: 53, Dst Port: 46749
◢ Domain Name System (response)
    Transaction ID: 0xbcab
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 21
  ▷ Queries
  ◢ Answers
    ▷ conf-cn.dc.heytapmobi.com: type A, class IN, addr 39.155.182.227
  ▷ Authoritative nameservers

0000  00 00 02 08 00 00 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 04 01 d1 a4 06 40 00  30 11 f0 7b 78 c4 a5 07   E·····@· 0··{x···
```

DNS 封包过滤



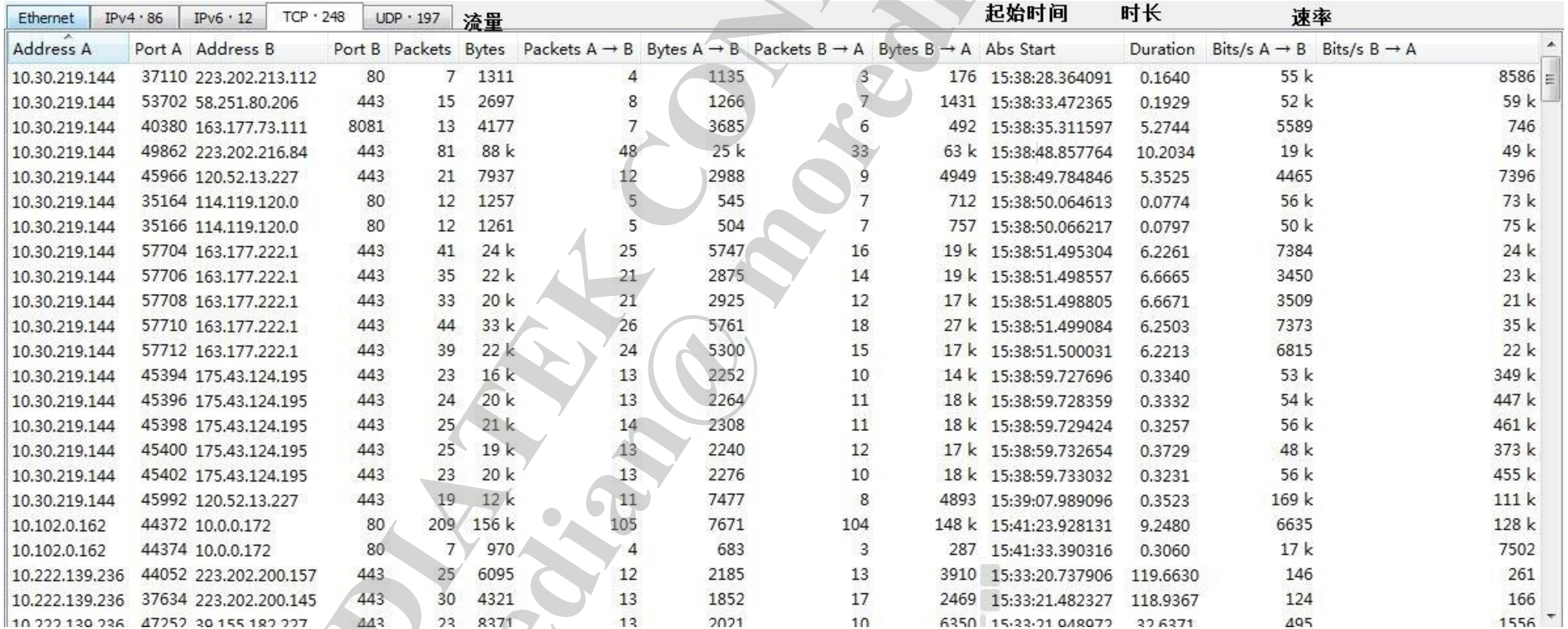
```
No.        Time                       Source                  Destination        Protocol  Length Info
      9253 2019/306 15:38:28.156483 fe80::1:2:bc40:be8c     ff02::2            ICMPv6        72 Router Solicitation from 1a:75:33:a0:f5:11
      9255 2019/306 15:38:28.401943 fe80::5                 fe80::1:2:bc40…    ICMPv6       104 Router Advertisement
      9262 2019/306 15:38:30.851927 fe80::5                 ff02::1            ICMPv6       104 Router Advertisement
      9791 2019/306 15:39:11.548494 fe80::1:1:6c53:a0e8     ff02::2            ICMPv6        72 Router Solicitation from 26:e7:dc:6f:55:ab
      9799 2019/306 15:39:11.862060 fe80::5                 fe80::1:1:6c53…    ICMPv6       104 Router Advertisement

▷ Frame 9799: 104 bytes on wire (832 bits), 104 bytes captured (832 bits)
▷ Linux cooked capture
▷ Internet Protocol Version 6, Src: fe80::5, Dst: fe80::1:1:6c53:a0e8
▷ Internet Control Message Protocol v6
```

ICMP 报文过滤

Wireshark 常用分析图表-Conversation

Conversation会按TCP、UDP、IPv4、IPv6等分类统计出抓取的封包，通过Conversation可以看到某一条流的流量，起始时间，时长，速率等。

菜单栏 统计>> 会话

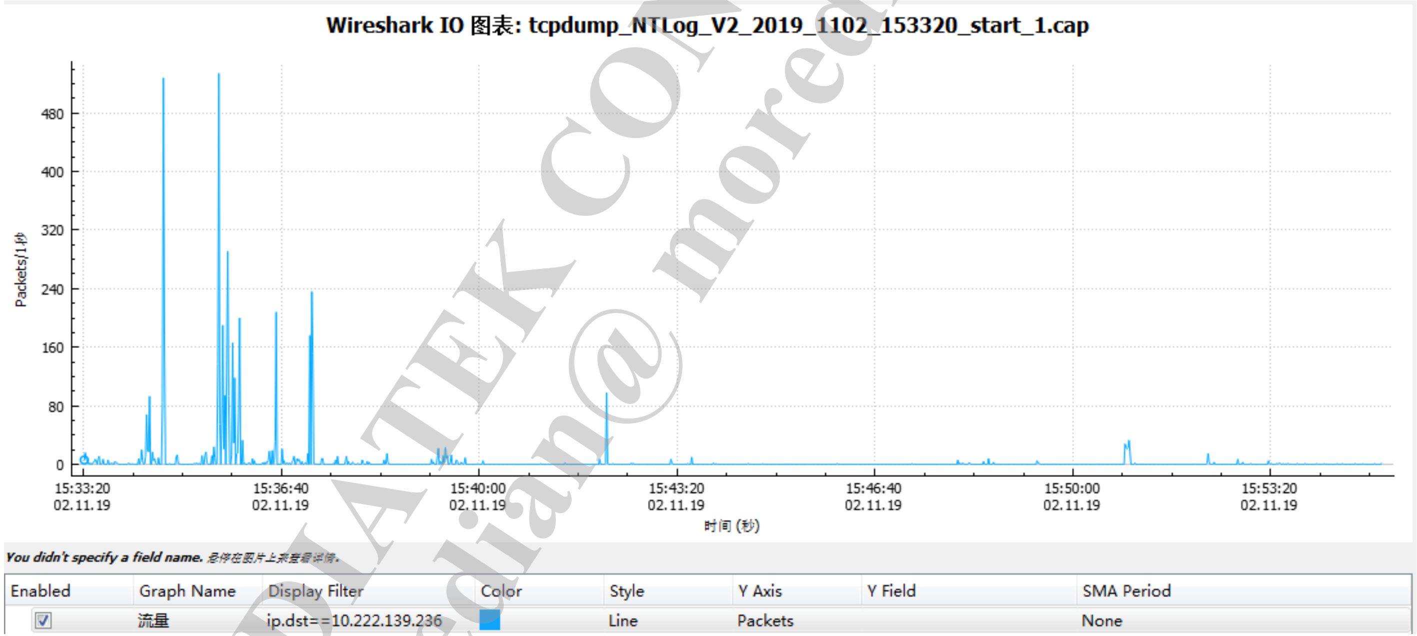


Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Abs Start	Duration	Bits/s A → B	Bits/s B → A
10.30.219.144	37110	223.202.213.112	80	7	1311	4	1135	3	176	15:38:28.364091	0.1640	55 k	8586
10.30.219.144	53702	58.251.80.206	443	15	2697	8	1266	7	1431	15:38:33.472365	0.1929	52 k	59 k
10.30.219.144	40380	163.177.73.111	8081	13	4177	7	3685	6	492	15:38:35.311597	5.2744	5589	746
10.30.219.144	49862	223.202.216.84	443	81	88 k	48	25 k	33	63 k	15:38:48.857764	10.2034	19 k	49 k
10.30.219.144	45966	120.52.13.227	443	21	7937	12	2988	9	4949	15:38:49.784846	5.3525	4465	7396
10.30.219.144	35164	114.119.120.0	80	12	1257	5	545	7	712	15:38:50.064613	0.0774	56 k	73 k
10.30.219.144	35166	114.119.120.0	80	12	1261	5	504	7	757	15:38:50.066217	0.0797	50 k	75 k
10.30.219.144	57704	163.177.222.1	443	41	24 k	25	5747	16	19 k	15:38:51.495304	6.2261	7384	24 k
10.30.219.144	57706	163.177.222.1	443	35	22 k	21	2875	14	19 k	15:38:51.498557	6.6665	3450	23 k
10.30.219.144	57708	163.177.222.1	443	33	20 k	21	2925	12	17 k	15:38:51.498805	6.6671	3509	21 k
10.30.219.144	57710	163.177.222.1	443	44	33 k	26	5761	18	27 k	15:38:51.499084	6.2503	7373	35 k
10.30.219.144	57712	163.177.222.1	443	39	22 k	24	5300	15	17 k	15:38:51.500031	6.2213	6815	22 k
10.30.219.144	45394	175.43.124.195	443	23	16 k	13	2252	10	14 k	15:38:59.727696	0.3340	53 k	349 k
10.30.219.144	45396	175.43.124.195	443	24	20 k	13	2264	11	18 k	15:38:59.728359	0.3332	54 k	447 k
10.30.219.144	45398	175.43.124.195	443	25	21 k	14	2308	11	18 k	15:38:59.729424	0.3257	56 k	461 k
10.30.219.144	45400	175.43.124.195	443	25	19 k	13	2240	12	17 k	15:38:59.732654	0.3729	48 k	373 k
10.30.219.144	45402	175.43.124.195	443	23	20 k	13	2276	10	18 k	15:38:59.733032	0.3231	56 k	455 k
10.30.219.144	45992	120.52.13.227	443	19	12 k	11	7477	8	4893	15:39:07.989096	0.3523	169 k	111 k
10.102.0.162	44372	10.0.0.172	80	209	156 k	105	7671	104	148 k	15:41:23.928131	9.2480	6635	128 k
10.102.0.162	44374	10.0.0.172	80	7	970	4	683	3	287	15:41:33.390316	0.3060	17 k	7502
10.222.139.236	44052	223.202.200.157	443	25	6095	12	2185	13	3910	15:33:20.737906	119.6630	146	261
10.222.139.236	37634	223.202.200.145	443	30	4321	13	1852	17	2469	15:33:21.482327	118.9367	124	166
10.222.139.236	47252	39.155.182.227	443	23	8371	13	2021	10	6350	15:33:21.948972	32.6371	495	1556

Wireshark 常用分析图表-IO Grapha

IO graphs是一个非常好用的工具。基本的Wireshark IO graph会显示抓包文件中的整体流量情况，通常是以每秒为单位（报文数或字节数）。

菜单栏　统计>> I/O图表



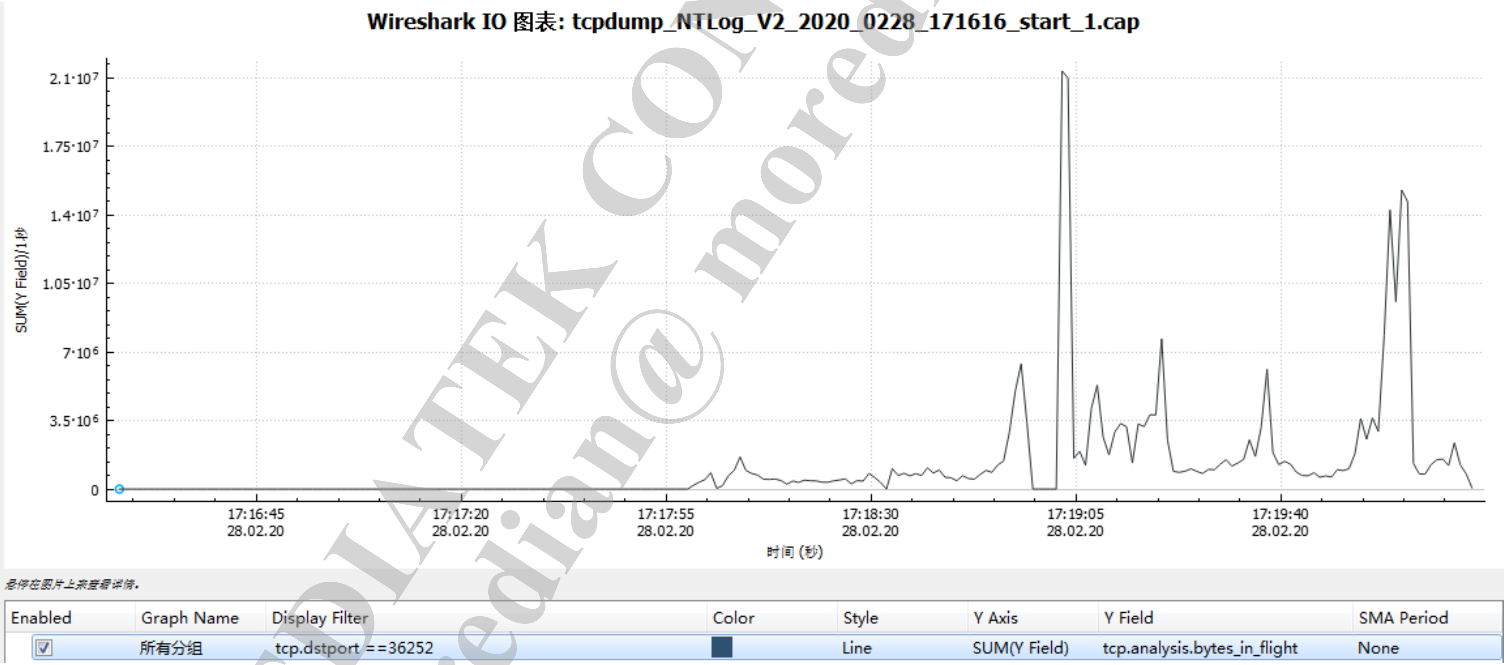
You didn't specify a field name. 悬停在图片上来查看详情.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	流量	ip.dst==10.222.139.236	🟦	Line	Packets		None

Wireshark 常用分析图表-IO Grapha

- **常用排错过滤条件:**

tcp.analysis.bytes_in_flight：某一时间点网络上未确认字节数。未确认字节数不能超过你的TCP窗口大小（定义于最初3此TCP握手），为了最大化吞吐量你想要获得尽可能接近TCP窗口大小。如果看到连续低于TCP窗口大小，可能意味着报文丢失或路径上其他影响吞吐量的问题。



Wireshark IO 图表: tcpdump_NTLog_V2_2020_0228_171616_start_1.cap

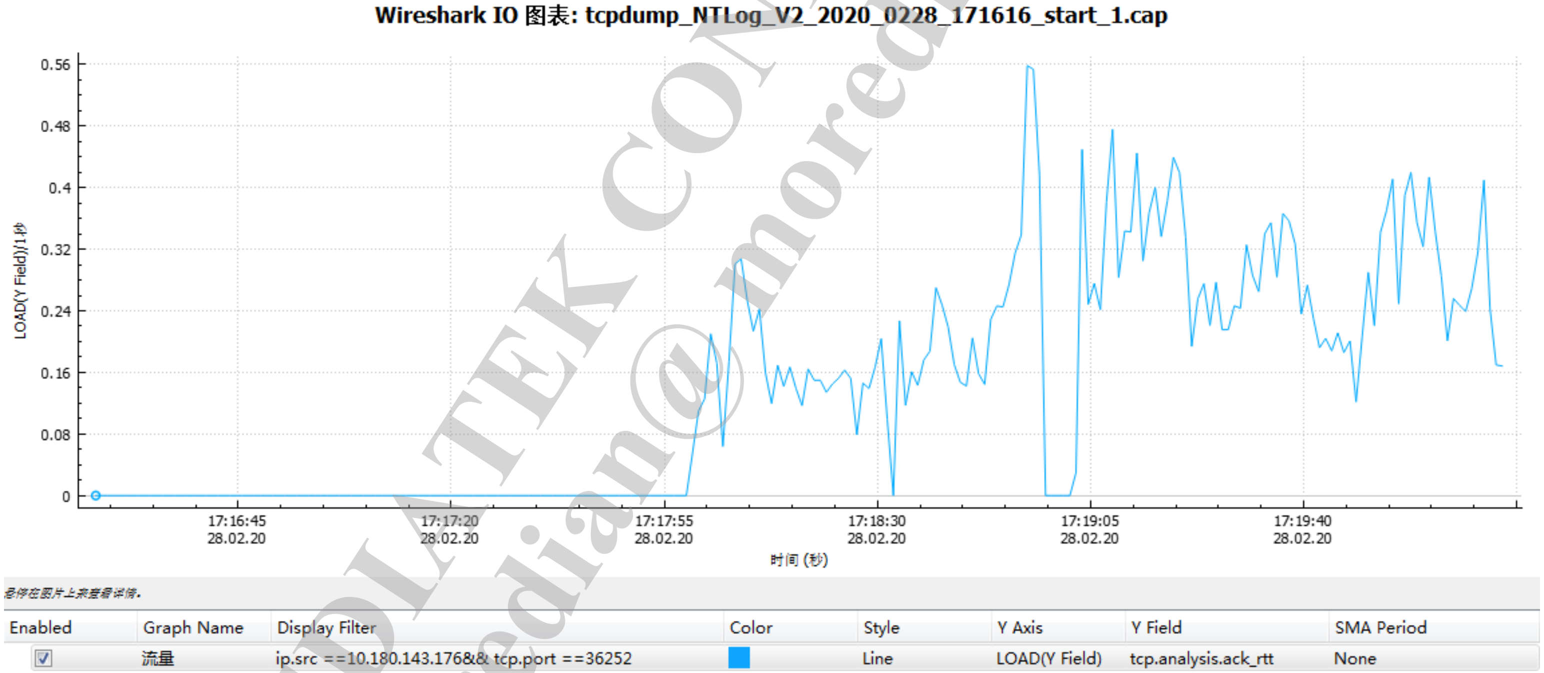
悬停在图片上来查看详情。

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	所有分组	tcp.dstport ==36252	■	Line	SUM(Y Field)	tcp.analysis.bytes_in_flight	None

Wireshark 常用分析图表-IO Grapha

- 常用排错过滤条件:

tcp.analysis.ack_rtt：衡量抓取的TCP报文与相应的ACK。如果这一时间间隔比较长那可能表示某种类型的网络延时（报文丢失，拥塞，等等）



悬停在图片上来查看详情。

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	流量	ip.src ==10.180.143.176&& tcp.port ==36252	🟦	Line	LOAD(Y Field)	tcp.analysis.ack_rtt	None

Wireshark 常用分析图表-IO Grapha

- **常用排错过滤条件:**

tcp.analysis.duplicate_ack：显示被确认过不止一次的报文。大凉的重复ACK是TCP端点之间高延时的迹象。

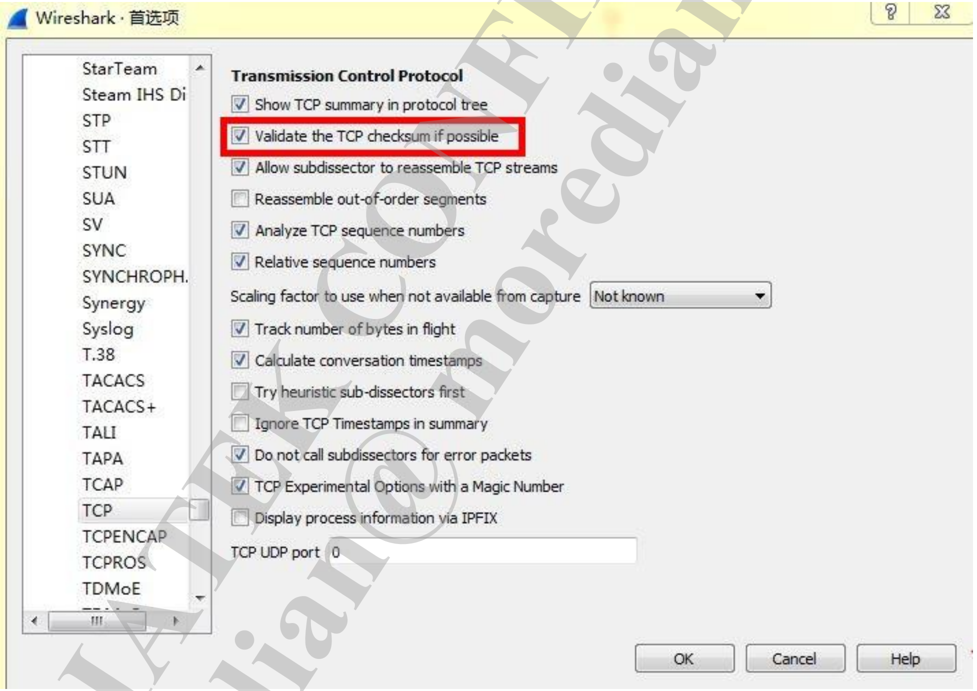
tcp.analysis.retransmission：显示抓包中的所有重传。如果重传次数不多的话还是正常的，过多重传可能有问题。这通常意味着应用性能缓慢和/或用户报文丢失。

tcp.analysis.window_update：将传输过程中的TCP window大小图形化。如果看到窗口大小下降为零，这意味着发送方已经退出了，并等待接收方确认所有已传送数据。这可能表明接收端已经不堪重负了。

tcp.analysis.lost_segment：表明已经在抓包中看到不连续的序列号。报文丢失会造成重复的ACK，这会导致重传。

Wireshark checksum 开启

菜单栏 编辑》首选项》Protocol》TCP



Transmission Control Protocol

☑ Show TCP summary in protocol tree

☑ Validate the TCP checksum if possible

☑ Allow subdissector to reassemble TCP streams

□ Reassemble out-of-order segments

☑ Analyze TCP sequence numbers

☑ Relative sequence numbers

Scaling factor to use when not available from capture: Not known

☑ Track number of bytes in flight

☑ Calculate conversation timestamps

□ Try heuristic sub-dissectors first

□ Ignore TCP Timestamps in summary

☑ Do not call subdissectors for error packets

☑ TCP Experimental Options with a Magic Number

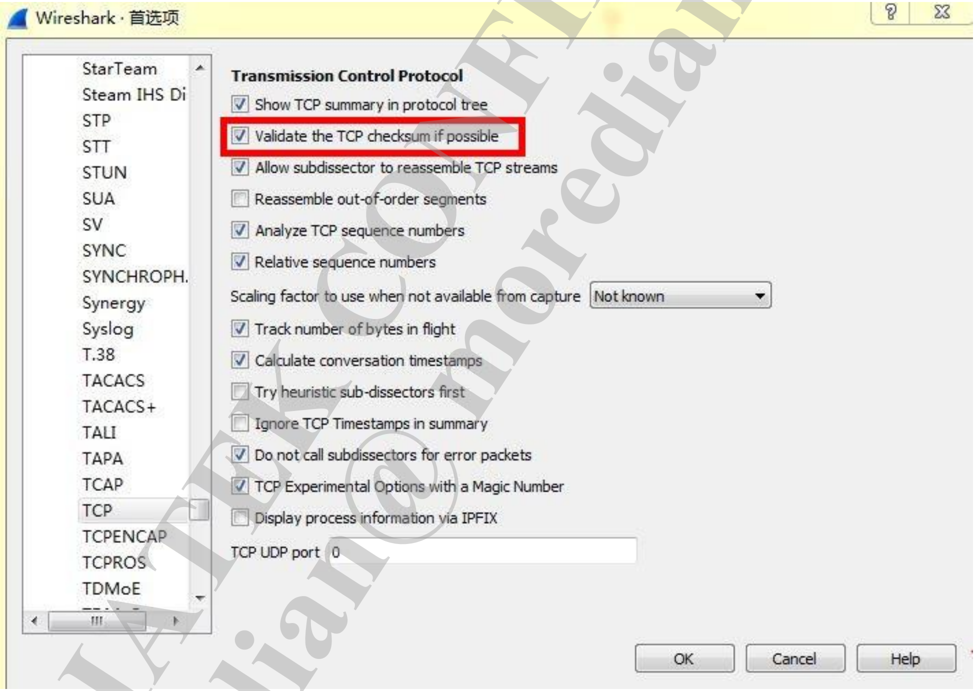
□ Display process information via IPFIX

TCP UDP port: 0

勾选validate the TCP checksum if possible, 这样wireshark 就会自动检测出checksum错误的数据包

Wireshark checksum 开启

菜单栏 编辑》首选项》Protocol》TCP



Transmission Control Protocol

- ☑ Show TCP summary in protocol tree
- ☑ Validate the TCP checksum if possible
- ☑ Allow subdissector to reassemble TCP streams
- ☐ Reassemble out-of-order segments
- ☑ Analyze TCP sequence numbers
- ☑ Relative sequence numbers

Scaling factor to use when not available from capture: Not known

- ☑ Track number of bytes in flight
- ☑ Calculate conversation timestamps
- ☐ Try heuristic sub-dissectors first
- ☐ Ignore TCP Timestamps in summary
- ☑ Do not call subdissectors for error packets
- ☑ TCP Experimental Options with a Magic Number
- ☐ Display process information via IPFIX

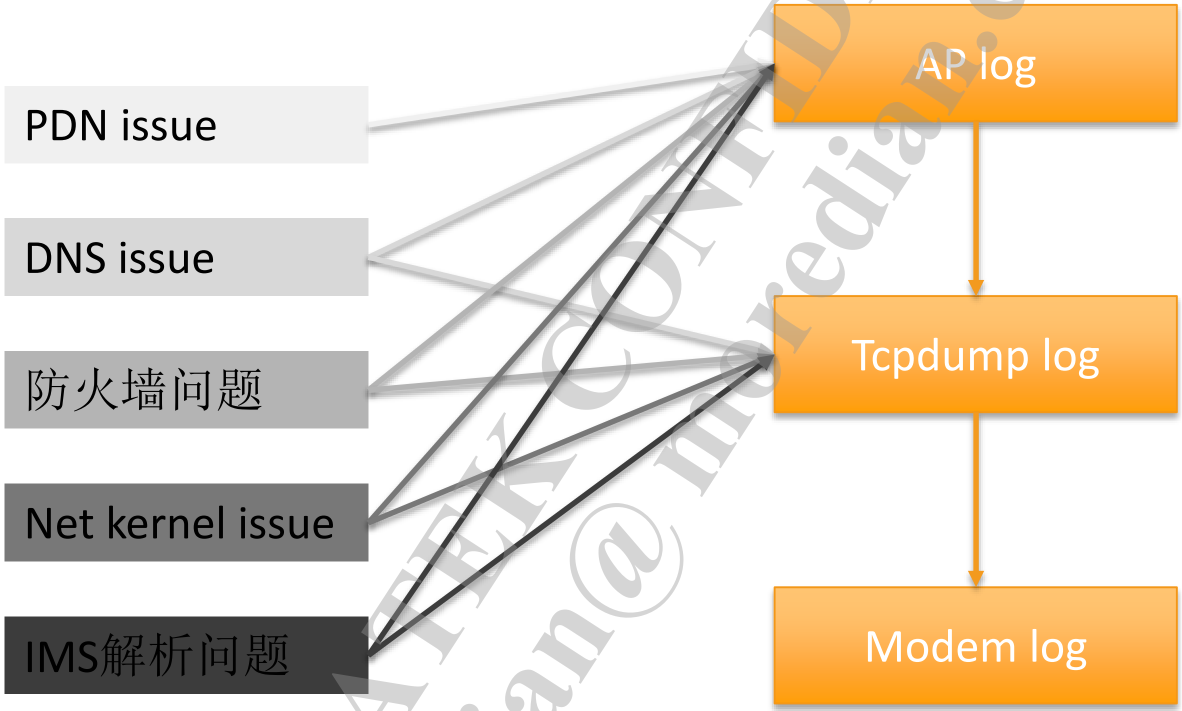
TCP UDP port: 0

勾选validate the TCP checksum if possible, 这样wireshark 就会自动检测出checksum错误的数据包

典型case 分析

纸上得来终觉浅

网络问题剖析

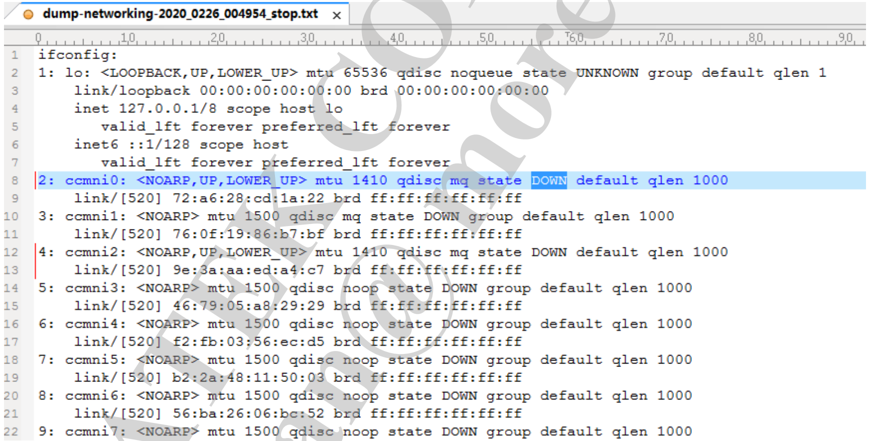

PDN issue
DNS issue
防火墙问题
Net kernel issue
IMS解析问题
AP log
Tcpdump log
Modem log

PDN issue

测试机能够访问网络的先决条件是数据连接建立起来（或连接上wifi，wifi连接在此不讨论）。

问题特征

- 问题发生时间段，netlog文件目录中的dump-networking_xxx.txt网口都是down状态，这种状态百分百是PDN建立异常issue



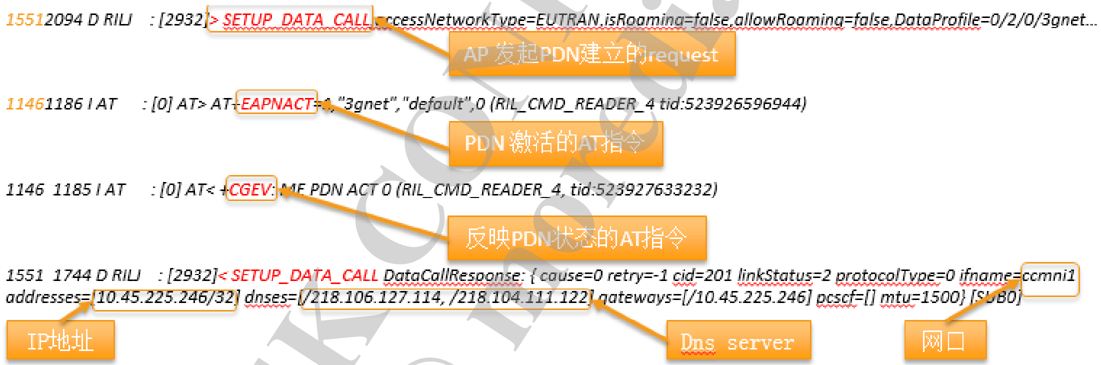
```
ifconfig:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ccmni0: <NOARP,UP,LOWER_UP> mtu 1410 qdisc mq state DOWN default qlen 1000
    link/[520] 72:a6:28:cd:1a:22 brd ff:ff:ff:ff:ff:ff
3: ccmni1: <NOARP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/[520] 76:0f:19:86:b7:bf brd ff:ff:ff:ff:ff:ff
4: ccmni2: <NOARP,UP,LOWER_UP> mtu 1410 qdisc mq state DOWN default qlen 1000
    link/[520] 9e:3a:aa:ed:a4:c7 brd ff:ff:ff:ff:ff:ff
5: ccmni3: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/[520] 46:79:05:a8:29:29 brd ff:ff:ff:ff:ff:ff
6: ccmni4: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/[520] f2:fb:03:56:ec:d5 brd ff:ff:ff:ff:ff:ff
7: ccmni5: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/[520] b2:2a:48:11:50:03 brd ff:ff:ff:ff:ff:ff
8: ccmni6: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/[520] 56:ba:26:06:bc:52 brd ff:ff:ff:ff:ff:ff
9: ccmni7: <NOARP> mtu 1500 qdisc noop state DOWN group default qlen 1000
```

- 问题发生时间段， netlog文件目录中tcpdump.cap没有数据包交互，或只有127.0.0.1的数据包交互， 这样需要进一步check 是防火墙丢包还是PDN建立异常问题

PDN issue

– 问题检测

- 对于PDN issue， 一般都是check AP log中的radio 和sys log
- Check 的关键词



```
15512094 D RILJ    : [2932]> SETUP_DATA_CALL,ccessNetworkType=EUTRAN,isRoaming=false,allowRoaming=false,DataProfile=0/2/0/3gnet...

11461186 I AT      : [0] AT> AT+EAPNACT=1,"3gnet","default",0 (RIL_CMD_READER_4 tid:523926596944)

1146  1185 I AT     : [0] AT< +CGEV: ME PDN ACT 0 (RIL_CMD_READER_4, tid:523927633232)

1551  1744 D RILJ    : [2932]< SETUP_DATA_CALL DataCallResponse: { cause=0 retry=-1 cid=201 linkStatus=2 protocolType=0 ifname=ccmni1
addresses=[10.45.225.246/32] dnses=[/218.106.127.114, /218.104.111.122] gateways=[/10.45.225.246] pcscf=[] mtu=1500} [SUB0]
```

– 常见问题

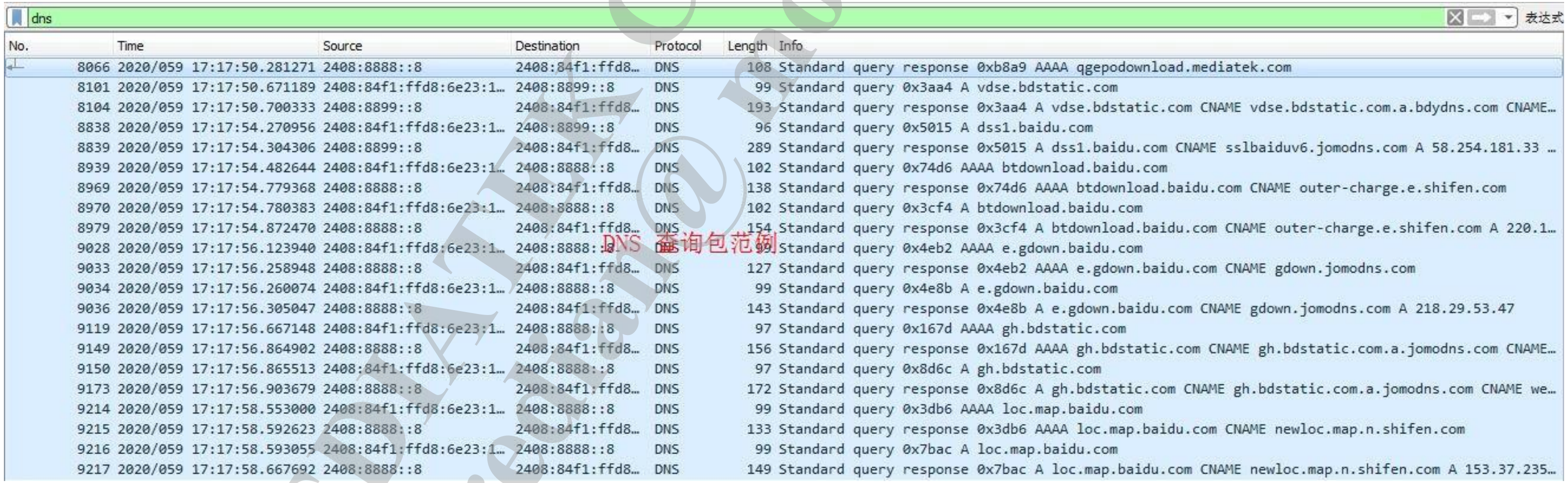
- 数据开关：没有开启；开关状态值FWK与UI不同步
- Apncontext状态值问题：譬如dataenable值为false
- Apn 配置问题：apn没有配置
- 漫游和漫游开关问题
- eapnact激活失败

DNS issue

DNS查询是访问网络第一步，只有通过DNS 查询获取到server的IP地址，测试机才能真正发起网络链接。

问题特征

- PDN建立正常，netlog文件目录中的dump-networking_xxx.txt也没有drop或reject记录，tcpdump_xxx.cap文件里看不到DNS 查询数据包，也看不到其他数据包。这种场景比较大可能是DNS查询包没有发出
- Tcpdump_xxx.cap中的DNS 查询包没有响应或没有返回IP地址



```
No.     Time                        Source                      Destination        Protocol  Length  Info
   8066 2020/059 17:17:50.281271 2408:8888::8                2408:84f1:ffd8…  DNS     108 Standard query response 0xb8a9 AAAA qgepodownload.mediatek.com
   8101 2020/059 17:17:50.671189 2408:84f1:ffd8:6e23:1…  2408:8899::8     DNS      99 Standard query 0x3aa4 A vdse.bdstatic.com
   8104 2020/059 17:17:50.700333 2408:8899::8                2408:84f1:ffd8…  DNS     193 Standard query response 0x3aa4 A vdse.bdstatic.com CNAME vdse.bdstatic.com.a.bdydns.com CNAME…
   8838 2020/059 17:17:54.270956 2408:84f1:ffd8:6e23:1…  2408:8899::8     DNS      96 Standard query 0x5015 A dss1.baidu.com
   8839 2020/059 17:17:54.304306 2408:8899::8                2408:84f1:ffd8…  DNS     289 Standard query response 0x5015 A dss1.baidu.com CNAME sslbaiduv6.jomodns.com A 58.254.181.33 …
   8939 2020/059 17:17:54.482644 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS     102 Standard query 0x74d6 AAAA btdownload.baidu.com
   8969 2020/059 17:17:54.779368 2408:8888::8                2408:84f1:ffd8…  DNS     138 Standard query response 0x74d6 AAAA btdownload.baidu.com CNAME outer-charge.e.shifen.com
   8970 2020/059 17:17:54.780383 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS     102 Standard query 0x3cf4 A btdownload.baidu.com
   8979 2020/059 17:17:54.872470 2408:8888::8                2408:84f1:ffd8…  DNS     154 Standard query response 0x3cf4 A btdownload.baidu.com CNAME outer-charge.e.shifen.com A 220.1…
   9028 2020/059 17:17:56.123940 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      99 Standard query 0x4eb2 AAAA e.gdown.baidu.com
   9033 2020/059 17:17:56.258948 2408:8888::8                2408:84f1:ffd8…  DNS     127 Standard query response 0x4eb2 AAAA e.gdown.baidu.com CNAME gdown.jomodns.com
   9034 2020/059 17:17:56.260074 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      99 Standard query 0x4e8b A e.gdown.baidu.com
   9036 2020/059 17:17:56.305047 2408:8888::8                2408:84f1:ffd8…  DNS     143 Standard query response 0x4e8b A e.gdown.baidu.com CNAME gdown.jomodns.com A 218.29.53.47
   9119 2020/059 17:17:56.667148 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      97 Standard query 0x167d AAAA gh.bdstatic.com
   9149 2020/059 17:17:56.864902 2408:8888::8                2408:84f1:ffd8…  DNS     156 Standard query response 0x167d AAAA gh.bdstatic.com CNAME gh.bdstatic.com.a.jomodns.com CNAME…
   9150 2020/059 17:17:56.865513 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      97 Standard query 0x8d6c A gh.bdstatic.com
   9173 2020/059 17:17:56.903679 2408:8888::8                2408:84f1:ffd8…  DNS     172 Standard query response 0x8d6c A gh.bdstatic.com CNAME gh.bdstatic.com.a.jomodns.com CNAME we…
   9214 2020/059 17:17:58.553000 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      99 Standard query 0x3db6 AAAA loc.map.baidu.com
   9215 2020/059 17:17:58.592623 2408:8888::8                2408:84f1:ffd8…  DNS     133 Standard query response 0x3db6 AAAA loc.map.baidu.com CNAME newloc.map.n.shifen.com
   9216 2020/059 17:17:58.593055 2408:84f1:ffd8:6e23:1…  2408:8888::8     DNS      99 Standard query 0x7bac A loc.map.baidu.com
   9217 2020/059 17:17:58.667692 2408:8888::8                2408:84f1:ffd8…  DNS     149 Standard query response 0x7bac A loc.map.baidu.com CNAME newloc.map.n.shifen.com A 153.37.235…
```

DNS 查询包范例

DNS issue

CASE 1 ：DNS包发送不出去

wifi和LTE切换，应用绑定的 netId没有及时更新， 与default netId不一致导致DNS查询包无法送出去，即netlog看不到DNS查询包

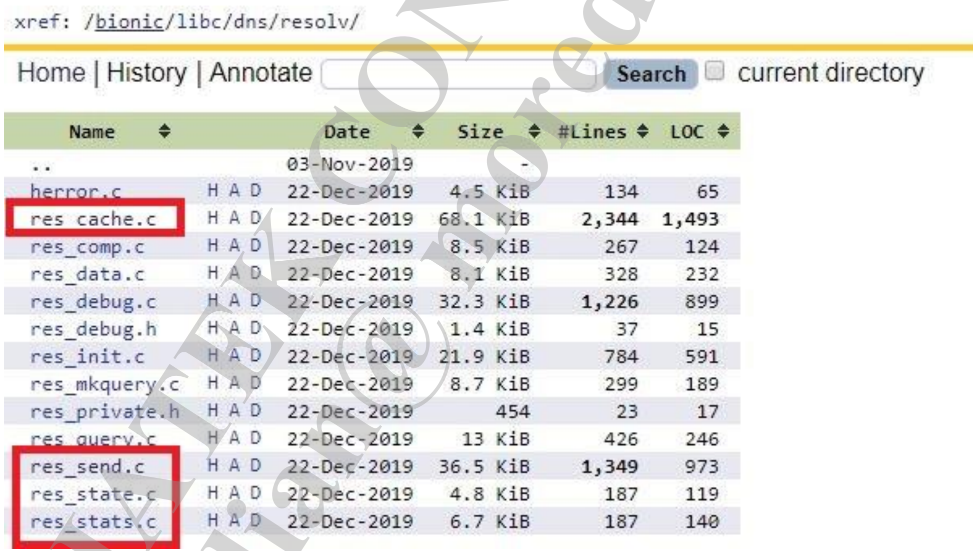
```
01-20 10:00:07.969607  1204  1475 D MtkConnectivityService: NetworkAgentInfo [WIFI () - 106] EVENT_NETWORK_INFO_CHANGED, going from CONNECTING to CONNECTED
01-20 10:00:08.970489  2950  8234 D Linux   : Linux_android_getaddrinfo netId =105,host =r1---sn-gwpa-qxak.gvt1.com
01-20 10:00:08.971067  2950  8234 D app_process64: android_getaddrinfofornetcontext netid = 105
01-20 10:00:08.971285  2950  8234 D app_process64: android_getaddrinfo_proxy netId = 105
01-20 10:00:08.971380   505   519 D DnsProxyListener: argv[0]=getaddrinfo
01-20 10:00:08.971416   505   519 D DnsProxyListener: argv[1]=r1---sn-gwpa-qxak.gvt1.com
01-20 10:00:08.971434   505   519 D DnsProxyListener: argv[2]=^
01-20 10:00:08.971451   505   519 D DnsProxyListener: argv[3]=1024
01-20 10:00:08.971468   505   519 D DnsProxyListener: argv[4]=0
01-20 10:00:08.971484   505   519 D DnsProxyListener: argv[5]=1
01-20 10:00:08.971500   505   519 D DnsProxyListener: argv[6]=0
01-20 10:00:08.971516   505   519 D DnsProxyListener: argv[7]=105
01-20 10:00:08.971545   505   519 D Netd    : app_netid:0x69 app_mark:0xf0069 dns_netid:0x6a dns_mark:0xe006a uid:10010
01-20 10:00:08.971570   505   519 D DnsProxyListener: GetAddrInfoHandler for r1---sn-gwpa-qxak.gvt1.com / [nullservice] / {105,983145,106,917610,10010}
01-20 10:00:08.971776   505  8330 D DnsProxyListener: GetAddrInfoHandler, now for r1---sn-gwpa-qxak.gvt1.com / (null) / {105,983145,106,917610,10010,0}
01-20 10:00:08.971904   505  8330 D netd    : android_getaddrinfofornetcontext netid = 105
01-20 10:00:08.972724  2950  8234 D libc-netbsd: [getaddrinfo]:  hostname=r1---sn-gwpa-qxak.gvt1.com;  servname=(null);  app_pid=2950;  app_uid=10010;  ai_flags=1024;  ai_family=0; ai_socktype=1 from prox result 7
```

DNS issue

– CASE 1 DNS包发送不出去

因为DNS相关的log现在默认都不打印，因此，这题分析起来比较困难，需要打开DNS log开关以及手动添加一些debug trace才能定位

打开DNS log开关的方法，将如下file中的DBG和DEBUG 变量由0改为1



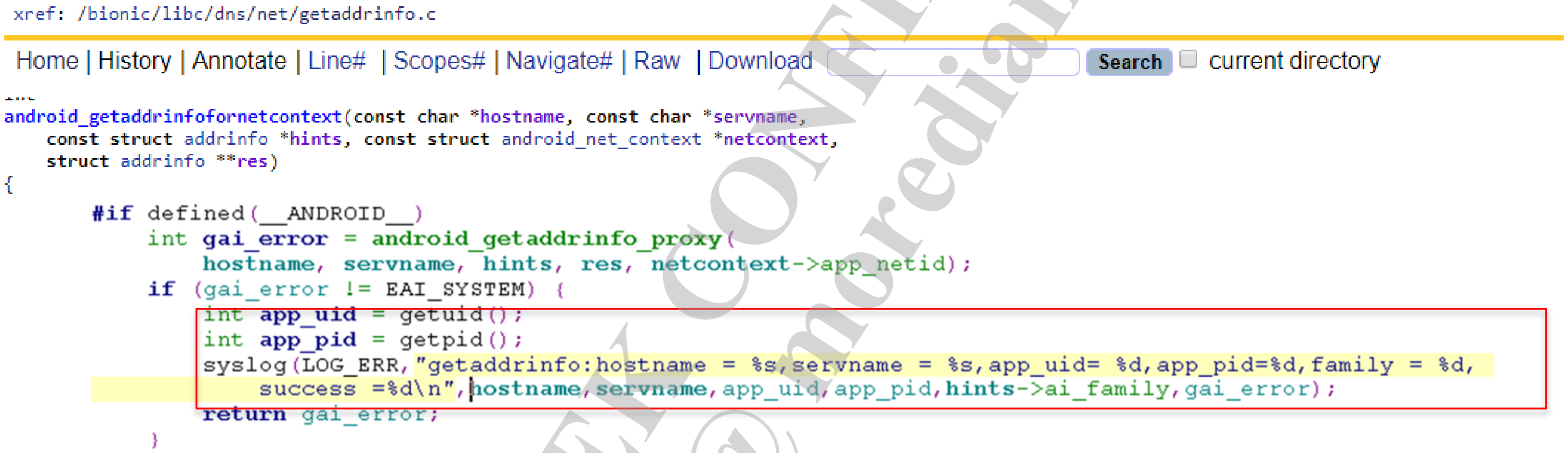
xref: /bionic/libc/dns/resolv/

Name		Date	Size	#Lines	LOC
..		03-Nov-2019	-		
herror.c	H A D	22-Dec-2019	4.5 KiB	134	65
res_cache.c	H A D	22-Dec-2019	68.1 KiB	2,344	1,493
res_comp.c	H A D	22-Dec-2019	8.5 KiB	267	124
res_data.c	H A D	22-Dec-2019	8.1 KiB	328	232
res_debug.c	H A D	22-Dec-2019	32.3 KiB	1,226	899
res_debug.h	H A D	22-Dec-2019	1.4 KiB	37	15
res_init.c	H A D	22-Dec-2019	21.9 KiB	784	591
res_mkquery.c	H A D	22-Dec-2019	8.7 KiB	299	189
res_private.h	H A D	22-Dec-2019	454	23	17
res_query.c	H A D	22-Dec-2019	13 KiB	426	246
res_send.c	H A D	22-Dec-2019	36.5 KiB	1,349	973
res_state.c	H A D	22-Dec-2019	4.8 KiB	187	119
res_stats.c	H A D	22-Dec-2019	6.7 KiB	187	140

DNS issue

– CASE 1 DNS包发送不出去

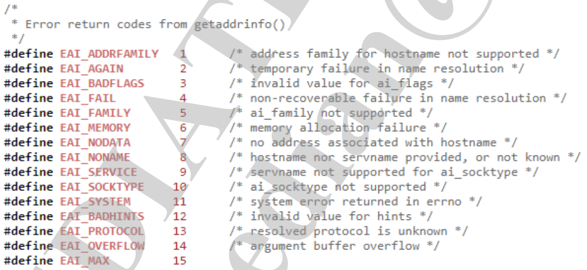
DNS查询结果debug trace添加



xref: /bionic/libc/dns/net/getaddrinfo.c

Home | History | Annotate | Line# | Scopes# | Navigate# | Raw | Download
Search
□ current directory

```
int
android_getaddrinfofornetcontext(const char *hostname, const char *servname,
    const struct addrinfo *hints, const struct android_net_context *netcontext,
    struct addrinfo **res)
{

      #if defined(__ANDROID__)
          int gai_error = android_getaddrinfo_proxy(
              hostname, servname, hints, res, netcontext->app_netid);
          if (gai_error != EAI_SYSTEM) {
              int app_uid = getuid();
              int app_pid = getpid();
              syslog(LOG_ERR,"getaddrinfo:hostname = %s,servname = %s,app_uid= %d,app_pid=%d,family = %d,
                  success =%d\n",hostname,servname,app_uid,app_pid,hints->ai_family,gai_error);
              return gai_error;
          }
```



```
/*
 * Error return codes from getaddrinfo()
 */
#define EAI_ADDRFAMILY   1      /* address family for hostname not supported */
#define EAI_AGAIN        2      /* temporary failure in name resolution */
#define EAI_BADFLAGS     3      /* invalid value for ai_flags */
#define EAI_FAIL         4      /* non-recoverable failure in name resolution */
#define EAI_FAMILY       5      /* ai_family not supported */
#define EAI_MEMORY       6      /* memory allocation failure */
#define EAI_NODATA       7      /* no address associated with hostname */
#define EAI_NONAME       8      /* hostname nor servname provided, or not known */
#define EAI_SERVICE      9      /* servname not supported for ai_socktype */
#define EAI_SOCKTYPE    10      /* ai_socktype not supported */
#define EAI_SYSTEM      11      /* system error returned in errno */
#define EAI_BADHINTS    12      /* invalid value for hints */
#define EAI_PROTOCOL    13      /* resolved protocol is unknown */
#define EAI_OVERFLOW    14      /* argument buffer overflow */
#define EAI_MAX         15
```

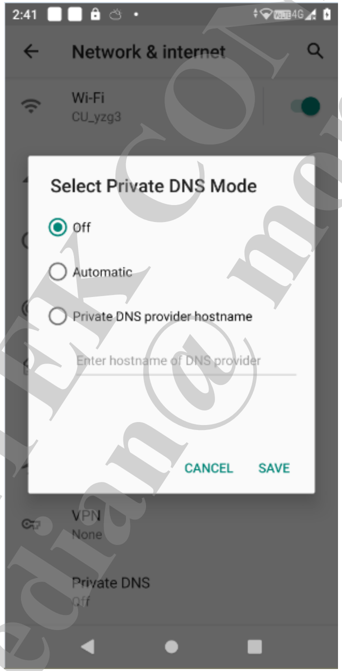
DNS查询返回错误类型

DNS issue

– CASE 2 ： private DNS issue

手机开启private DNS之后，netlog中看不到DNS查询包，DNS 查询返回失败。

开启private dns之后，DNS查询包通过加密的方式送出，默认使用的端口为853（正常DNS查询包的DNS 端口为53）。由于当前大部分网络运营商还不支持private DNS server ,因此，DNS 查询会失败



2:41

Network & internet

Wi-Fi
CU_yzg3

Select Private DNS Mode

● Off

○ Automatic

○ Private DNS provider hostname

Enter hostname of DNS provider

VPN
None

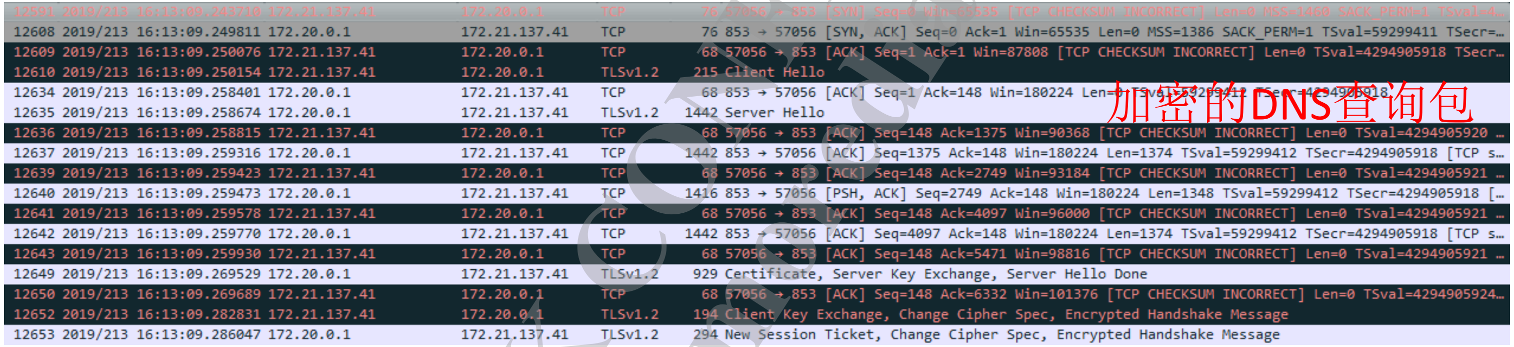
Private DNS
Off

- # DNS issue

  - CASE 2 private DNS issue

关键log

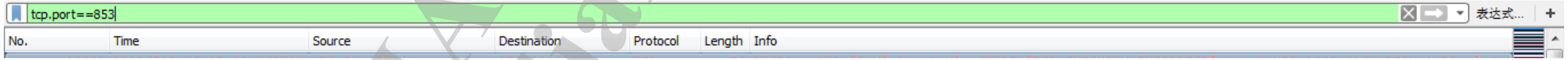
08-01 15:12:09.175576   913 1023 W DnsManager: updatePrivateDns(100, PrivateDnsConfig{true:/[]})



```
12591 2019/213 16:13:09.243710 172.21.137.41       172.20.0.1      TCP      76 57056 → 853 [SYN] Seq=0 Win=65535 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 SACK_PERM=1 TSval=4…
12608 2019/213 16:13:09.249811 172.20.0.1          172.21.137.41   TCP      76 853 → 57056 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 SACK_PERM=1 TSval=59299411 TSecr=…
12609 2019/213 16:13:09.250076 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=1 Ack=1 Win=87808 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905918 TSecr…
12610 2019/213 16:13:09.250154 172.21.137.41       172.20.0.1      TLSv1.2 215 Client Hello
12634 2019/213 16:13:09.258401 172.20.0.1          172.21.137.41   TCP      68 853 → 57056 [ACK] Seq=1 Ack=148 Win=180224 Len=0 TSval=59299412 TSecr=4294905918
12635 2019/213 16:13:09.258674 172.20.0.1          172.21.137.41   TLSv1.2 1442 Server Hello
12636 2019/213 16:13:09.258815 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=148 Ack=1375 Win=90368 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905920 …
12637 2019/213 16:13:09.259316 172.20.0.1          172.21.137.41   TCP    1442 853 → 57056 [ACK] Seq=1375 Ack=148 Win=180224 Len=1374 TSval=59299412 TSecr=4294905918 [TCP s…
12639 2019/213 16:13:09.259423 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=148 Ack=2749 Win=93184 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905921 …
12640 2019/213 16:13:09.259473 172.20.0.1          172.21.137.41   TCP    1416 853 → 57056 [PSH, ACK] Seq=2749 Ack=148 Win=180224 Len=1348 TSval=59299412 TSecr=4294905918 […
12641 2019/213 16:13:09.259578 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=148 Ack=4097 Win=96000 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905921 …
12642 2019/213 16:13:09.259770 172.20.0.1          172.21.137.41   TCP    1442 853 → 57056 [ACK] Seq=4097 Ack=148 Win=180224 Len=1374 TSval=59299412 TSecr=4294905918 [TCP s…
12643 2019/213 16:13:09.259930 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=148 Ack=5471 Win=98816 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905921 …
12649 2019/213 16:13:09.269529 172.20.0.1          172.21.137.41   TLSv1.2  929 Certificate, Server Key Exchange, Server Hello Done
12650 2019/213 16:13:09.269689 172.21.137.41       172.20.0.1      TCP      68 57056 → 853 [ACK] Seq=148 Ack=6332 Win=101376 [TCP CHECKSUM INCORRECT] Len=0 TSval=4294905924…
12652 2019/213 16:13:09.282831 172.21.137.41       172.20.0.1      TLSv1.2  194 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12653 2019/213 16:13:09.286047 172.20.0.1          172.21.137.41   TLSv1.2  294 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

加密的DNS查询包

Tip：如果Ap log中搜不到private dns 开启的关键log，netlog中也看不到对应的DNS查询包，可以在netlog 的filter中尝试搜索tcp.port == 853 试试

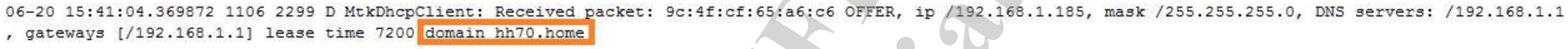


tcp.port==853

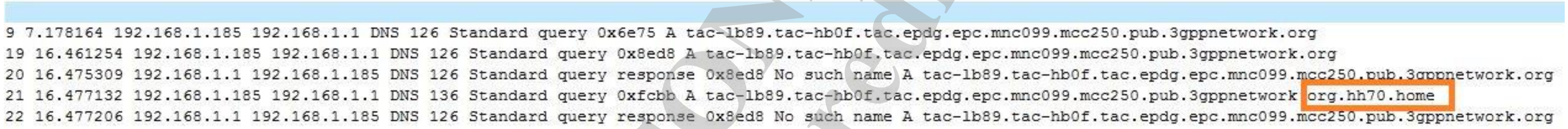
DNS issue

– CASE 3　DNS查询携带domain后缀

- 关键log



```
06-20 15:41:04.369872 1106 2299 D MtkDhcpClient: Received packet: 9c:4f:cf:65:a6:c6 OFFER, ip /192.168.1.185, mask /255.255.255.0, DNS servers: /192.168.1.1
, gateways [/192.168.1.1] lease time 7200 domain hh70.home
```

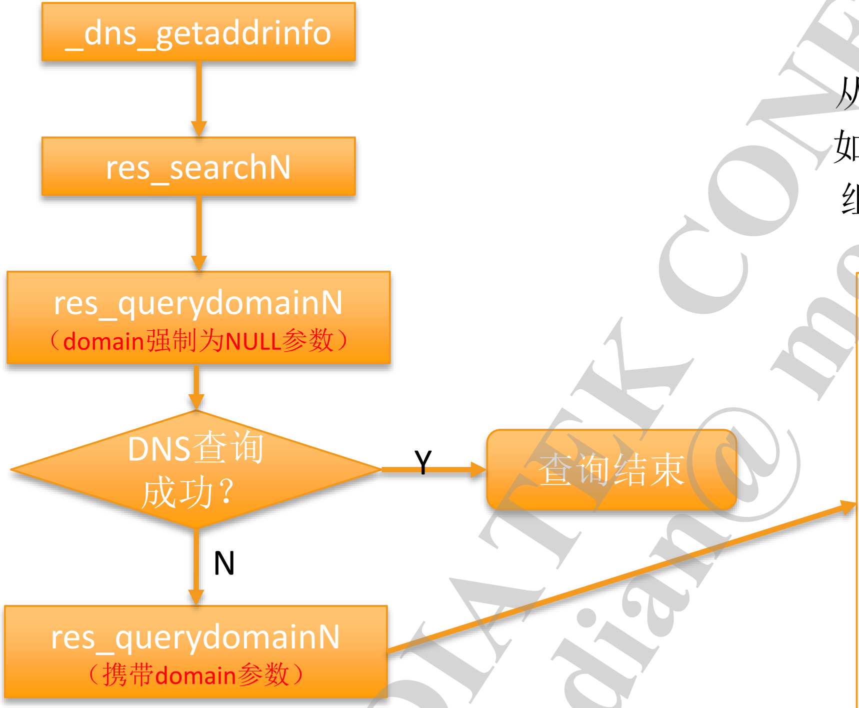


```
9 7.178164 192.168.1.185 192.168.1.1 DNS 126 Standard query 0x6e75 A tac-lb89.tac-hb0f.tac.epdg.epc.mnc099.mcc250.pub.3gppnetwork.org
19 16.461254 192.168.1.185 192.168.1.1 DNS 126 Standard query 0x8ed8 A tac-lb89.tac-hb0f.tac.epdg.epc.mnc099.mcc250.pub.3gppnetwork.org
20 16.475309 192.168.1.1 192.168.1.185 DNS 126 Standard query response 0x8ed8 No such name A tac-lb89.tac-hb0f.tac.epdg.epc.mnc099.mcc250.pub.3gppnetwork.org
21 16.477132 192.168.1.185 192.168.1.1 DNS 136 Standard query 0xfcbb A tac-lb89.tac-hb0f.tac.epdg.epc.mnc099.mcc250.pub.3gppnetwork org.hh70.home
22 16.477206 192.168.1.1 192.168.1.185 DNS 126 Standard query response 0x8ed8 No such name A tac-lb89.tac-hb0f.tac.epdg.epc.mnc099.mcc250.pub.3gppnetwork.org
```

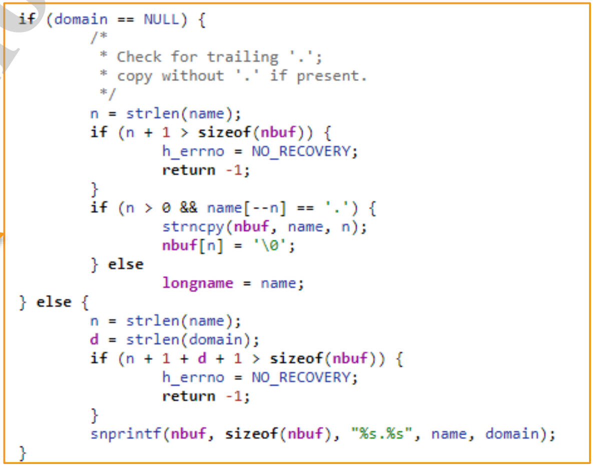
- 这是DNS查询的正常行为，当DNS查询没有返回结果，并且获取IP地址有拿到domain 信息，DNS查询就会尝试使用hostname+domain进行查询。譬如要查询的hostname是www.baidu.com, domain 是mediatek.com。如果查询www.baidu.com没有返回IP地址，DNS会自动执行www.baidu.com.mediate.com的查询

DNS issue

– CASE 3　DNS查询携带domain后缀



从下图的res_querydomainN 实现可以看到如果domain!=NULL,就会将hostname+domain 组合到一起进行查询



```
if (domain == NULL) {
        /*
         * Check for trailing '.';
         * copy without '.' if present.
         */
        n = strlen(name);
        if (n + 1 > sizeof(nbuf)) {
                h_errno = NO_RECOVERY;
                return -1;
        }
        if (n > 0 && name[--n] == '.') {
                strncpy(nbuf, name, n);
                nbuf[n] = '\0';
        } else
                longname = name;
} else {
        n = strlen(name);
        d = strlen(domain);
        if (n + 1 + d + 1 > sizeof(nbuf)) {
                h_errno = NO_RECOVERY;
                return -1;
        }
        snprintf(nbuf, sizeof(nbuf), "%s.%s", name, domain);
}
```

DNS issue

- CASE 3　DNS查询携带domain后缀

● 规避方案

1.修改RES_DEFAULT初始值，将RES_DNSRCH去掉

```
#define RES_DEFAULT (RES_RECURSE | RES_DEFNAMES | RES_DNSRCH | RES_NO_NIBBLE2)
```

File path:/bionic/libc/dns/include/resolv_private.h

2. 在设置DNS 参数时， 忽略掉domain 的赋值

```
public void setDnsConfigurationForNetwork( int netId, LinkProperties lp, boolean isDefaultNetwork) {
            final String[] assignedServers = NetworkUtils.makeStrings(lp.getDnsServers());
-           final String[] domainStrs = getDomainStrings(lp.getDomains());
+           final String[] domainStrs = getDomainStrings("");
```

File path:frameworks/base/services/core/java/com/android/server/connectivity/DnsManager.java

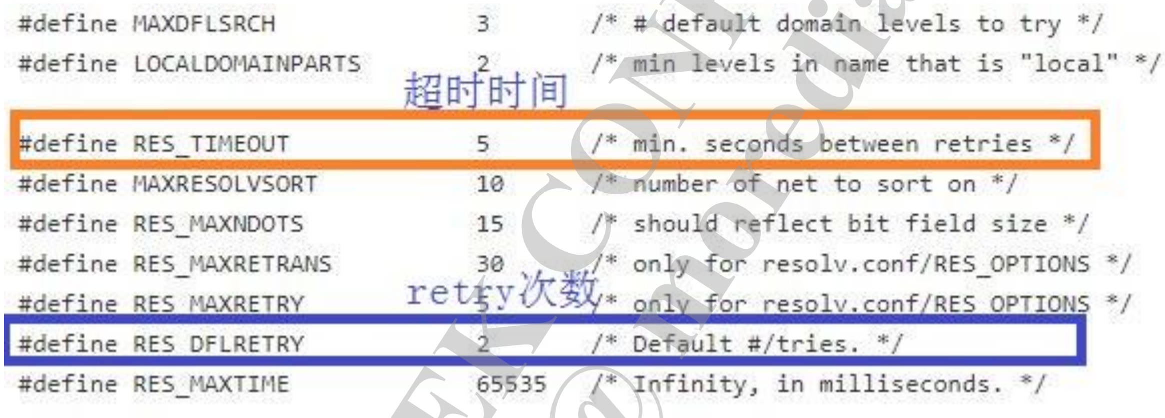
3. 在res_searchN方法中进行客制化处理，第一次DNS查询失败，不进行hostname+domain的查询

DNS issue

CASE 4 DNS常见客制化修改

- DNS 查询超时时间和重试次数修改

file path: /bionic/libc/dns/include/resolv_private.h



```
#define MAXDFLSRCH          3       /* # default domain levels to try */
#define LOCALDOMAINPARTS    2       /* min levels in name that is "local" */

#define RES_TIMEOUT         5       /* min. seconds between retries */
#define MAXRESOLVSORT       10      /* number of net to sort on */
#define RES_MAXNDOTS        15      /* should reflect bit field size */
#define RES_MAXRETRANS      30      /* only for resolv.conf/RES_OPTIONS */
#define RES_MAXRETRY        5       /* only for resolv.conf/RES_OPTIONS */
#define RES_DFLRETRY        2       /* Default #/tries. */
#define RES_MAXTIME         65535   /* Infinity, in milliseconds. */
```

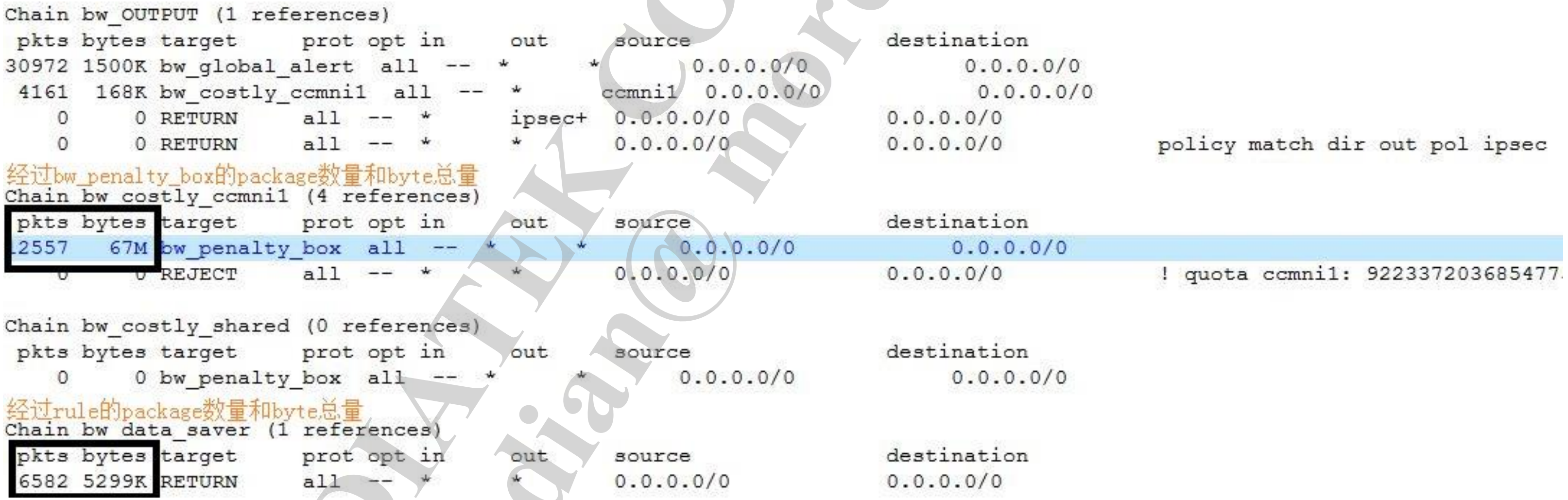
- IP 地址返回次序修改

这个在_rfc6724_sort方法中修改,涉及到rfc6724，因此，要谨慎处理

file path: /bionic/libc/dns/net/getaddrinfo.c

Firewall issue

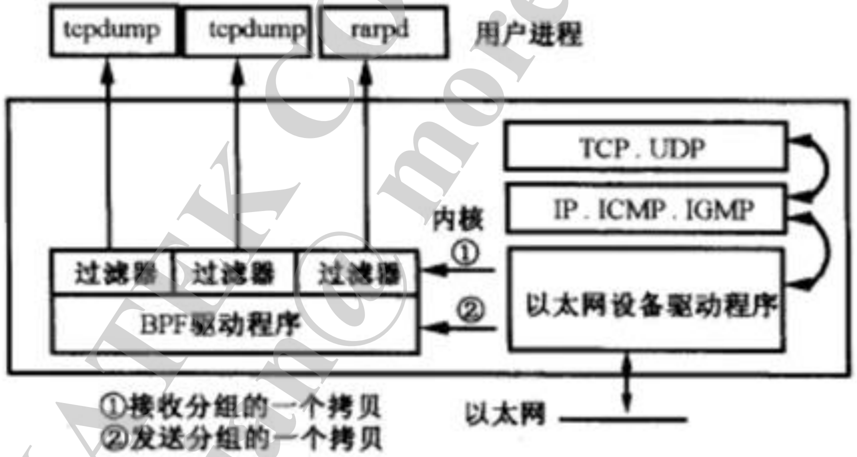
防火墙是网络分析中常见的一类问题，而且该类问题有一定的隐蔽性，不特意去check 防火墙rule ,一般还比较难发现。对于mtk log， 防火墙rule一般会在netlog目录下dump-networking-xxx.txt中打印（需要注意的一点是：这支文件需要先关闭log工具，再导出log，才能抓到）,在每条rule或chain的前面会记录经过其的package数目和bytes总量



```
Chain bw_OUTPUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
30972 1500K bw_global_alert  all  --  *      *       0.0.0.0/0            0.0.0.0/0
 4161  168K bw_costly_ccmni1  all  --  *      ccmni1  0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  *      ipsec+  0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0           policy match dir out pol ipsec
经过bw_penalty_box的package数量和byte总量
Chain bw_costly_ccmni1 (4 references)
 pkts bytes target     prot opt in     out     source               destination
12557   67M bw_penalty_box  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ! quota ccmni1: 92233720368547

Chain bw_costly_shared (0 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 bw_penalty_box  all  --  *      *       0.0.0.0/0            0.0.0.0/0
经过rule的package数量和byte总量
Chain bw_data_saver (1 references)
 pkts bytes target     prot opt in     out     source               destination
6582 5299K RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

- Firewall issue

我们知道防火墙kernel实现主要在 layer 3,即IP层。数据包的抓取工具是tcpdump，而tcpdump实现如图所示，主要是从网口驱动上拷贝一份数据。因此，如果有防火墙：发出的数据包，netlog看不到；收到的数据包，netlog可见，但用户层无法获取。**这是我们判断是否是防火墙丢包问题的一个重要准则**



Firewall issue

– CASE 1 禁止DNS查询的方式阻挡数据传输

工模>>Telephony中有一个Backgroud Data Select选项 ,该选项 的防火墙策略是只放行特定字符串的DNS查询包，其他的DNS 查询包会被drop 。其实现原理是通过禁用DNS查询端口53来实现，如果此时是private dns ,这各设置就不生效.why?

```
 Chain oem_data (1 references)
pkts bytes target     prot opt in     out     source          destination
 0     0 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0          udp dpt:53 STRING match  "3gppnetwork" ALGO name bm TO 65535
0     0 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0          udp dpt:53 STRING match  "slp.rs.de" ALGO name bm TO 65535
0     0 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0          udp dpt:53 STRING match  "spirent" ALGO name bm TO 65535
7864  504K DROP       udp  --  *      *       0.0.0.0/0          0.0.0.0/0          udp dpt:53
0     0 ACCEPT     all  --  *      *       0.0.0.0/0          1.2.3.4
0     0 ACCEPT     all  --  *      *       0.0.0.0/0          1.1.1.1
30  2270 ACCEPT     all  --  *      *       0.0.0.0/0          192.168.0.0/16
0     0 ACCEPT     all  --  *      *       0.0.0.0/0          172.16.0.0/12
0     0 ACCEPT     all  --  *      *       0.0.0.0/0          10.0.0.0/8
0     0 DROP       all  --  *      ppp+    0.0.0.0/0          0.0.0.0/0
```

Tip：快速定位防火墙问题的一个简单方式，是在netlog目录中的dump-networking-xxx.txt 文件中搜索关键字DROP 或REJECT ，如果前面的pkts和bytes栏位不为0，说明就有防火墙丢包发生

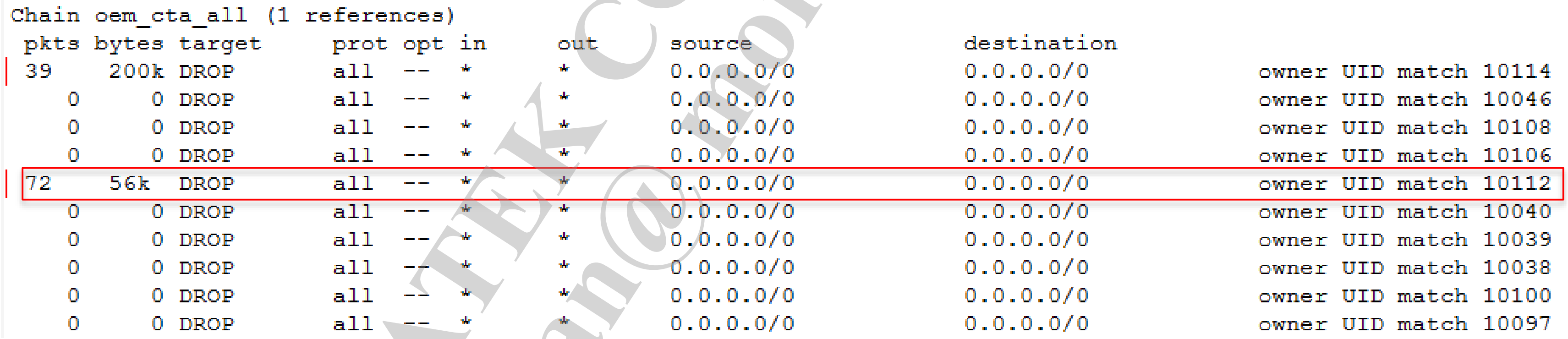
- ## Firewall issue

  - ### CASE 2 禁止APP UID方式阻挡APP数据传输

从下图可以看到UID为10114和10112的APP有丢包发生。在event log搜索UID可以找到其对应APP进程。

我们也可以反向查找，譬如用户反馈浏览器无法上网，其他上网应用正常，可以通过找到浏览器的UID为10112，然后在dump-networking-xxx.txt中搜索该10112，发现其有防火墙丢包，则其上不了网的问题很大可能是因为防火墙引起。

02-28 17:18:39.596251 1104 1158 I am_pss : [5624,10112,com.android.browser,135465984,125054976,143360,272695296,0,16,22]



```
Chain oem_cta_all (1 references)
 pkts bytes target     prot opt in     out     source               destination
 39    200k DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10114
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10046
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10108
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10106
 72    56k  DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10112
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10040
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10039
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10038
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10100
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            owner UID match 10097
```

TIP：对于防火墙问题，可以通过链名（譬如oem_cta_all）来判断出是MTK原生防火墙设计还是客制化，来决定下一步处理

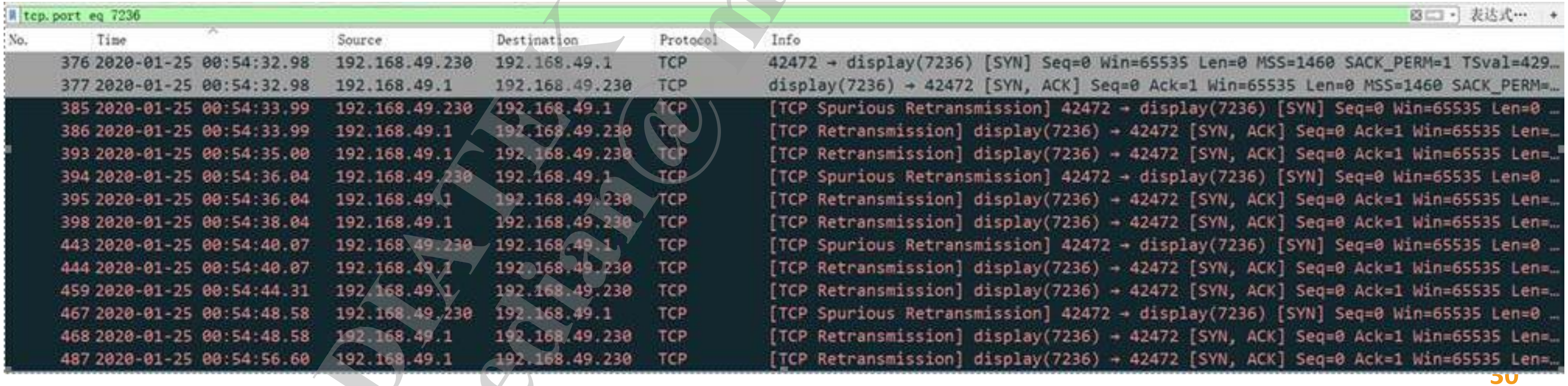
Network kernel issue

network kernel 性能整体稳定，一般很少出题。这里讲的kernel issue是指由于network driver或网络端传递过来数据包参数有异常，导致kernel直接将该包丢弃或断开数据连接.

正确配置wireshark 工具，wireshark 就可以自动识别出异常数据包

Case 1 checksum error

如下图所示，DUT一直在重传SYN包，但网络端有回syn ack 。从现象判断syn ack 似乎没有收到，怀疑syn ack 包有被drop

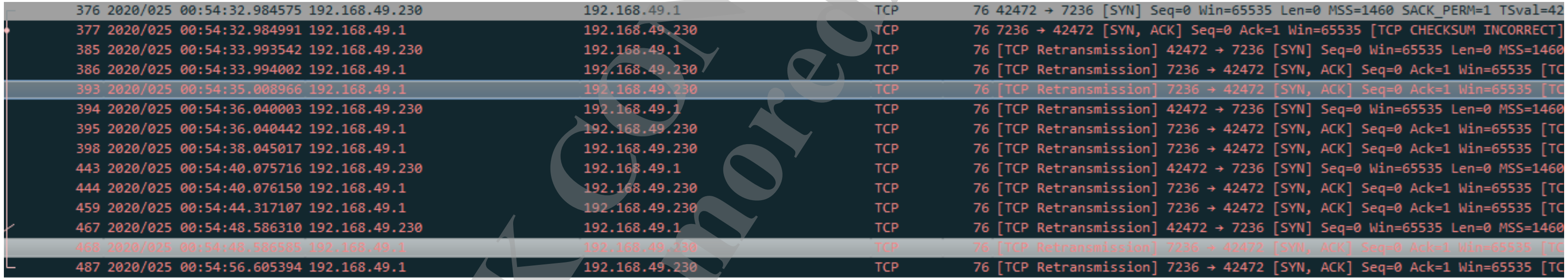


```
No.  Time                    Source          Destination     Protocol  Info
376 2020-01-25 00:54:32.98  192.168.49.230  192.168.49.1    TCP   42472 → display(7236) [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=429…
377 2020-01-25 00:54:32.98  192.168.49.1    192.168.49.230  TCP   display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=…
385 2020-01-25 00:54:33.99  192.168.49.230  192.168.49.1    TCP   [TCP Spurious Retransmission] 42472 → display(7236) [SYN] Seq=0 Win=65535 Len=0 …
386 2020-01-25 00:54:33.99  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
393 2020-01-25 00:54:35.00  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
394 2020-01-25 00:54:36.04  192.168.49.230  192.168.49.1    TCP   [TCP Spurious Retransmission] 42472 → display(7236) [SYN] Seq=0 Win=65535 Len=0 …
395 2020-01-25 00:54:36.04  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
398 2020-01-25 00:54:38.04  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
443 2020-01-25 00:54:40.07  192.168.49.230  192.168.49.1    TCP   [TCP Spurious Retransmission] 42472 → display(7236) [SYN] Seq=0 Win=65535 Len=0 …
444 2020-01-25 00:54:40.07  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
459 2020-01-25 00:54:44.31  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
467 2020-01-25 00:54:48.58  192.168.49.230  192.168.49.1    TCP   [TCP Spurious Retransmission] 42472 → display(7236) [SYN] Seq=0 Win=65535 Len=0 …
468 2020-01-25 00:54:48.58  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
487 2020-01-25 00:54:56.60  192.168.49.1    192.168.49.230  TCP   [TCP Retransmission] display(7236) → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=…
```

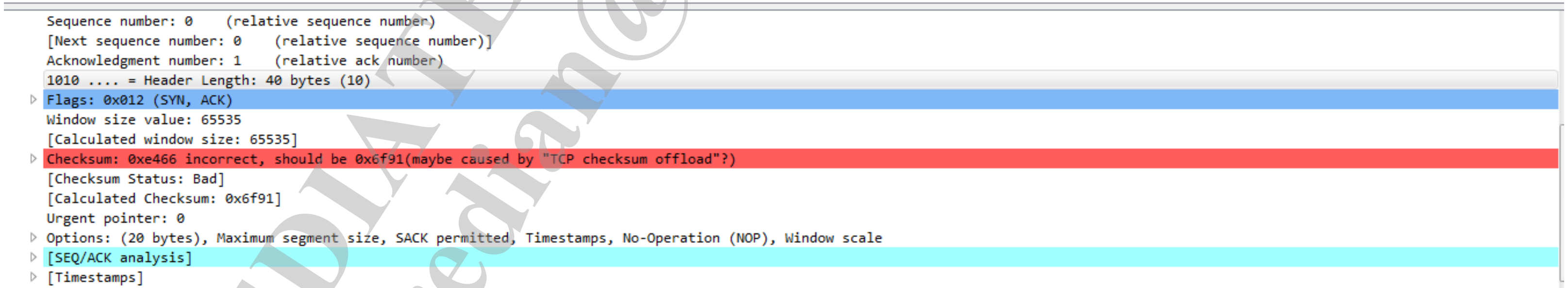
Network kernel issue

– Case 1 checksum error

如PPT第一章所述，将编辑》首选项》Protocol》TCP 中的validate the TCP checksum if possible 勾选（安装wireshark 时，这个选项默认没有勾选，建议勾选上）。选中syn ack数据包，在其封包详细信息，TCP选项中可以看到checksum eror。**收到的数据包如果checksum error ,kernel就会drop ;发出的数据包不用care checksum报错**



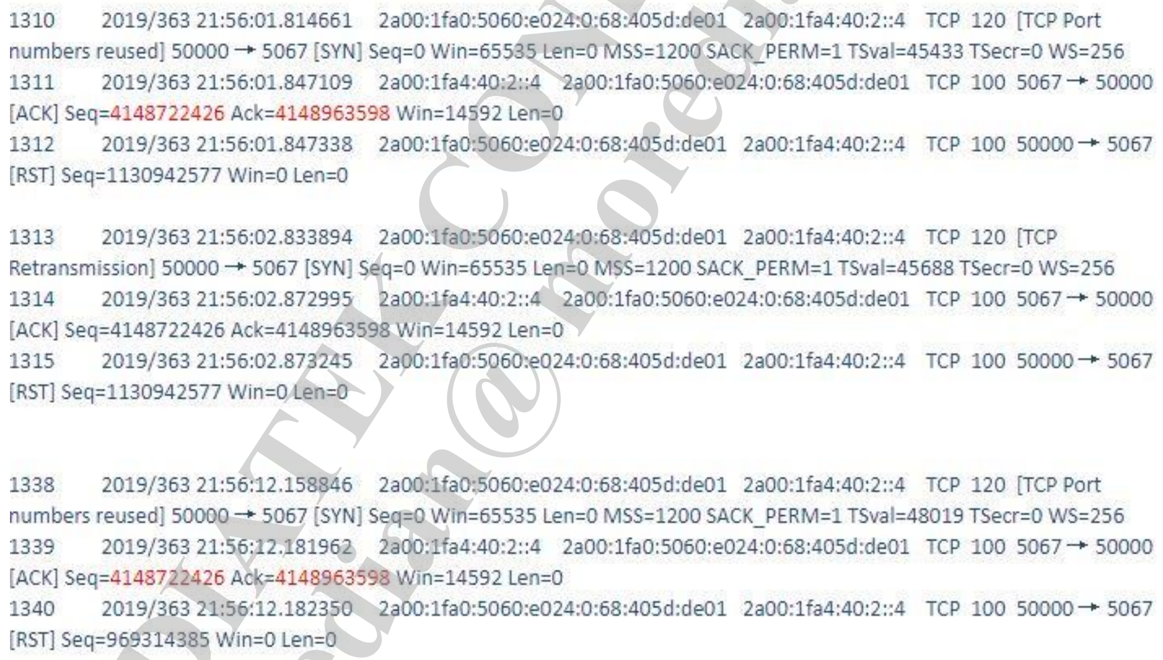
```
376 2020/025 00:54:32.984575 192.168.49.230        192.168.49.1         TCP   76 42472 → 7236 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=42
377 2020/025 00:54:32.984991 192.168.49.1          192.168.49.230       TCP   76 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TCP CHECKSUM INCORRECT]
385 2020/025 00:54:33.993542 192.168.49.230        192.168.49.1         TCP   76 [TCP Retransmission] 42472 → 7236 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
386 2020/025 00:54:33.994002 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
393 2020/025 00:54:35.008966 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
394 2020/025 00:54:36.040003 192.168.49.230        192.168.49.1         TCP   76 [TCP Retransmission] 42472 → 7236 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
395 2020/025 00:54:36.040442 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
398 2020/025 00:54:38.045017 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
443 2020/025 00:54:40.075716 192.168.49.230        192.168.49.1         TCP   76 [TCP Retransmission] 42472 → 7236 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
444 2020/025 00:54:40.076150 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
459 2020/025 00:54:44.317107 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
467 2020/025 00:54:48.586310 192.168.49.230        192.168.49.1         TCP   76 [TCP Retransmission] 42472 → 7236 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
468 2020/025 00:54:48.586585 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
487 2020/025 00:54:56.605394 192.168.49.1          192.168.49.230       TCP   76 [TCP Retransmission] 7236 → 42472 [SYN, ACK] Seq=0 Ack=1 Win=65535 [TC
```



```
  Sequence number: 0    (relative sequence number)
  [Next sequence number: 0    (relative sequence number)]
  Acknowledgment number: 1    (relative ack number)
  1010 .... = Header Length: 40 bytes (10)
▷ Flags: 0x012 (SYN, ACK)
  Window size value: 65535
  [Calculated window size: 65535]
▷ Checksum: 0xe466 incorrect, should be 0x6f91(maybe caused by "TCP checksum offload"?)
  [Checksum Status: Bad]
  [Calculated Checksum: 0x6f91]
  Urgent pointer: 0
▷ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
▷ [SEQ/ACK analysis]
▷ [Timestamps]
```

- Network kernel issue
  - Case 2 ack number erorr

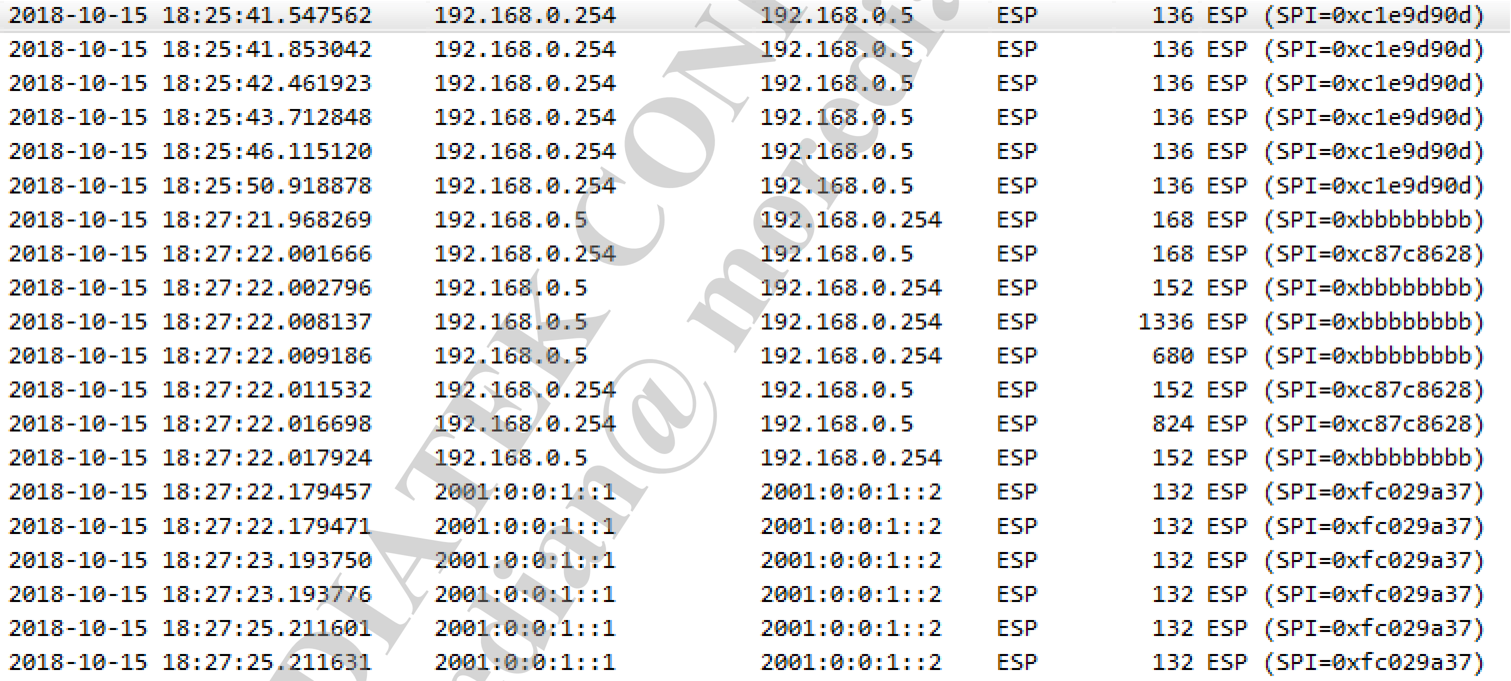
TCP 三次握手时，DUT发出syn ,网络端返回的syn ack number不对，kernel会直接发起RST，TCP连接无法建立



```
1310    2019/363 21:56:01.814661   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  120  [TCP Port
numbers reused] 50000 → 5067 [SYN] Seq=0 Win=65535 Len=0 MSS=1200 SACK_PERM=1 TSval=45433 TSecr=0 WS=256
1311    2019/363 21:56:01.847109   2a00:1fa4:40:2::4   2a00:1fa0:5060:e024:0:68:405d:de01   TCP  100  5067 → 50000
[ACK] Seq=4148722426 Ack=4148963598 Win=14592 Len=0
1312    2019/363 21:56:01.847338   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  100  50000 → 5067
[RST] Seq=1130942577 Win=0 Len=0

1313    2019/363 21:56:02.833894   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  120  [TCP
Retransmission] 50000 → 5067 [SYN] Seq=0 Win=65535 Len=0 MSS=1200 SACK_PERM=1 TSval=45688 TSecr=0 WS=256
1314    2019/363 21:56:02.872995   2a00:1fa4:40:2::4   2a00:1fa0:5060:e024:0:68:405d:de01   TCP  100  5067 → 50000
[ACK] Seq=4148722426 Ack=4148963598 Win=14592 Len=0
1315    2019/363 21:56:02.873245   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  100  50000 → 5067
[RST] Seq=1130942577 Win=0 Len=0


1338    2019/363 21:56:12.158846   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  120  [TCP Port
numbers reused] 50000 → 5067 [SYN] Seq=0 Win=65535 Len=0 MSS=1200 SACK_PERM=1 TSval=48019 TSecr=0 WS=256
1339    2019/363 21:56:12.181962   2a00:1fa4:40:2::4   2a00:1fa0:5060:e024:0:68:405d:de01   TCP  100  5067 → 50000
[ACK] Seq=4148722426 Ack=4148963598 Win=14592 Len=0
1340    2019/363 21:56:12.182350   2a00:1fa0:5060:e024:0:68:405d:de01   2a00:1fa4:40:2::4   TCP  100  50000 → 5067
[RST] Seq=969314385 Win=0 Len=0
```

IMS network issue

IMS network issue主要有两大类，一是连接无法建立；二是发出的SIP消息未正确响应。由于IMS 数据包需要加密处理，如下所示。因此，分析IMS网络问题关键的一步是能够**解析出IMS 加密数据包**



```
2018-10-15 18:25:41.547562   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:25:41.853042   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:25:42.461923   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:25:43.712848   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:25:46.115120   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:25:50.918878   192.168.0.254   192.168.0.5     ESP    136 ESP (SPI=0xc1e9d90d)
2018-10-15 18:27:21.968269   192.168.0.5     192.168.0.254   ESP    168 ESP (SPI=0xbbbbbbbb)
2018-10-15 18:27:22.001666   192.168.0.254   192.168.0.5     ESP    168 ESP (SPI=0xc87c8628)
2018-10-15 18:27:22.002796   192.168.0.5     192.168.0.254   ESP    152 ESP (SPI=0xbbbbbbbb)
2018-10-15 18:27:22.008137   192.168.0.5     192.168.0.254   ESP   1336 ESP (SPI=0xbbbbbbbb)
2018-10-15 18:27:22.009186   192.168.0.5     192.168.0.254   ESP    680 ESP (SPI=0xbbbbbbbb)
2018-10-15 18:27:22.011532   192.168.0.254   192.168.0.5     ESP    152 ESP (SPI=0xc87c8628)
2018-10-15 18:27:22.016698   192.168.0.254   192.168.0.5     ESP    824 ESP (SPI=0xc87c8628)
2018-10-15 18:27:22.017924   192.168.0.5     192.168.0.254   ESP    152 ESP (SPI=0xbbbbbbbb)
2018-10-15 18:27:22.179457   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
2018-10-15 18:27:22.179471   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
2018-10-15 18:27:23.193750   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
2018-10-15 18:27:23.193776   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
2018-10-15 18:27:25.211601   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
2018-10-15 18:27:25.211631   2001:0:0:1::1   2001:0:0:1::2   ESP    132 ESP (SPI=0xfc029a37)
```

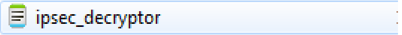
IMS network issue

IMS 解析工具使用

- Download IPSec_Decryptor

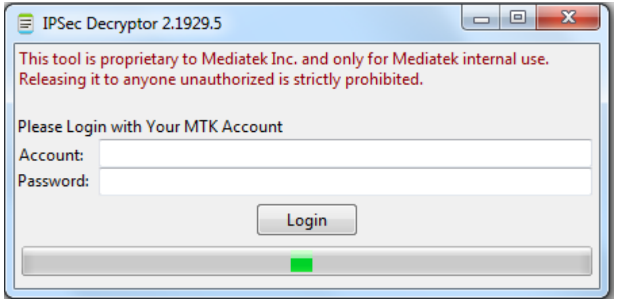
  Link: \\script.gslb.mediatek.inc\script\WirelessTools\ToolRelease\IPSec_Decryptor

- Execute ipsec_decryptor.exe



ipsec_decryptor

- Login
  - Make sure there is a MTK internal network connection on your PC. (Being able to access CQ system at least.)
  - Input your MTK account and password to login
    - Currently only authorized below teams to use this tool:
      - WSP
      - WSD
      - WCS
      - CTD



IPSec Decryptor 2.1929.5

This tool is proprietary to Mediatek Inc. and only for Mediatek internal use. Releasing it to anyone unauthorized is strictly prohibited.

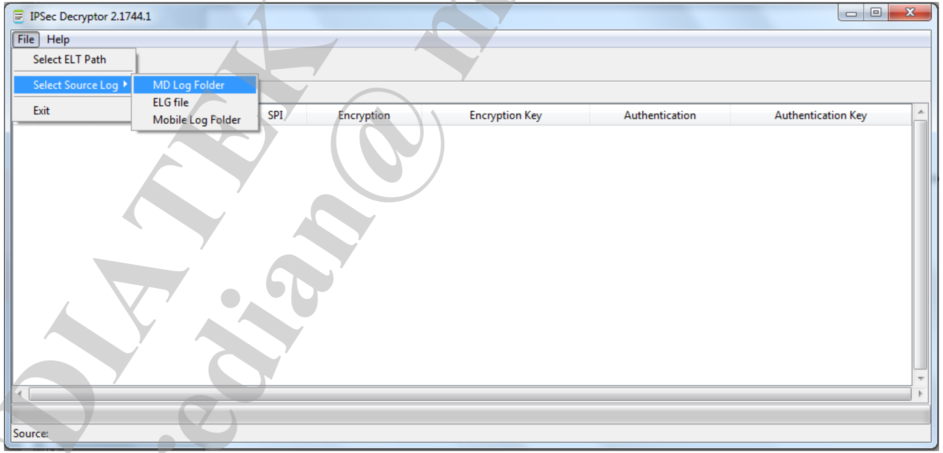
Please Login with Your MTK Account

Account:

Password:

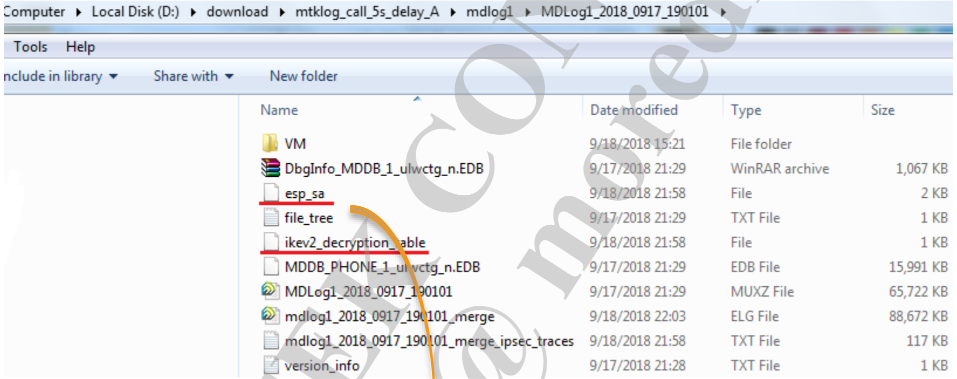
IMS network issue

- Fill in ELT Path & Source Log Path , then click green triangle button to start parsing.
  - If you select “MD Log Folder”, the muxraw/muxz file under the selected folder will be merged into a ELG log. MD Database file is also necessary for MDLogMan process in this step. Then tool will search the ELG log file and find out ESP/IKEv2 IPSec keys.
  - If you select “ELG file”, tool will search the ELG log file and find out ESP/IKEv2 IPSec keys.
  - If you select “Mobile Log Folder”, only the main_log file under the selected folder will be parsed. Tool will try to find out ESP/IKEv2 IPSec keys, but not always success because some SW loads won’t print related information in main_log files.



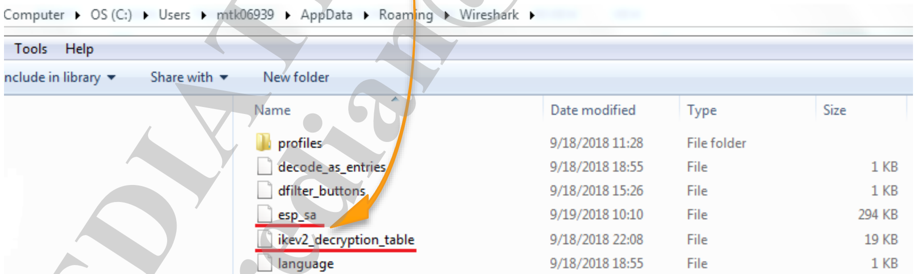
IMS network issue

- If IPSec key is found, esp_sa or ikev2_decryption_table will be generated to save Key
  - The parsed IPSec Key lines will also be appended to the esp_sa or ikev2_decryption_table file under “C:\Users\xxxxx\AppData\Roaming\Wireshark\”.



Computer ▸ Local Disk (D:) ▸ download ▸ mtklog_call_5s_delay_A ▸ mdlog1 ▸ MDLog1_2018_0917_190101 ▸

Name	Date modified	Type	Size
VM	9/18/2018 15:21	File folder	
DbgInfo_MDDB_1_ulwctg_n.EDB	9/17/2018 21:29	WinRAR archive	1,067 KB
esp_sa	9/18/2018 21:58	File	2 KB
file_tree	9/17/2018 21:29	TXT File	1 KB
ikev2_decryption_table	9/18/2018 21:58	File	1 KB
MDDB_PHONE_1_ulwctg_n.EDB	9/17/2018 21:29	EDB File	15,991 KB
MDLog1_2018_0917_190101	9/17/2018 21:29	MUXZ File	65,722 KB
mdlog1_2018_0917_190101_merge	9/18/2018 22:03	ELG File	88,672 KB
mdlog1_2018_0917_190101_merge_ipsec_traces	9/18/2018 21:58	TXT File	117 KB
version_info	9/17/2018 21:28	TXT File	1 KB



Computer ▸ OS (C:) ▸ Users ▸ mtk06939 ▸ AppData ▸ Roaming ▸ Wireshark ▸

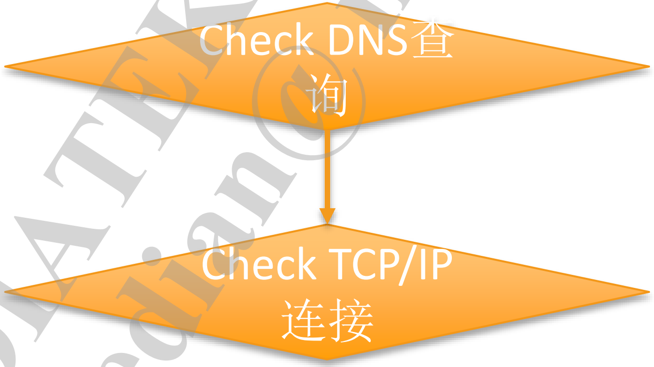
Name	Date modified	Type	Size
profiles	9/18/2018 11:28	File folder	
decode_as_entries	9/18/2018 18:55	File	1 KB
dfilter_buttons	9/18/2018 15:26	File	1 KB
esp_sa	9/19/2018 10:10	File	294 KB
ikev2_decryption_table	9/18/2018 22:08	File	19 KB
language	9/18/2018 18:55	File	1 KB

Third app network access issue

Network performance是我们网络分析中常见的一大类问题，主要表现为网络访问异常，譬如网页刷新慢，界面提示网络未连接，游戏卡顿等

因为这类问题，PDN连接是正常的，所以对其分析就需要借助于网络分析工具wireshark，第一章中介绍到的conversation,IO Grapha功能此时就会排上用场。

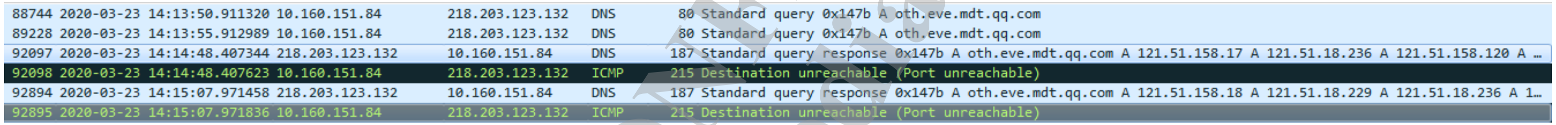
该类问题的分析模型如下



Third app network access issue

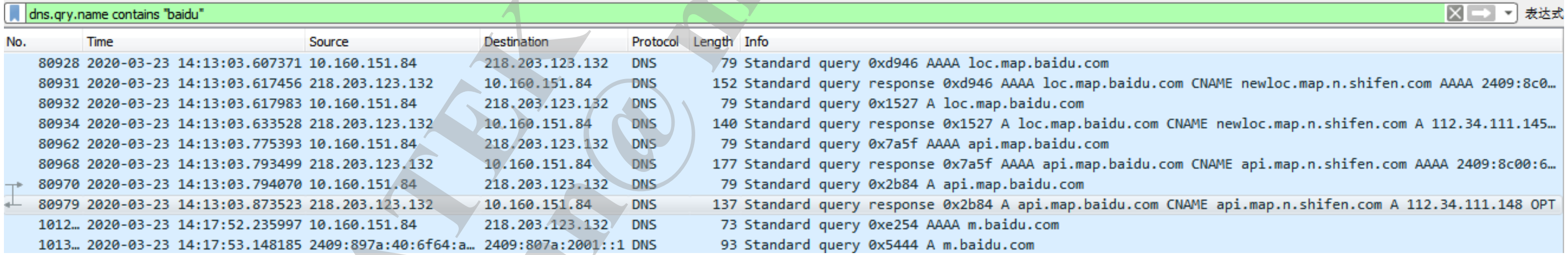
– Case 1 DNS查询延时

Dns查询默认的超时时间是5秒。如果5秒之后结果才返回，查询的socket可能已经关闭，该结果就无效，如下图所示



```
88744 2020-03-23 14:13:50.911320 10.160.151.84        218.203.123.132  DNS        80 Standard query 0x147b A oth.eve.mdt.qq.com
89228 2020-03-23 14:13:55.912989 10.160.151.84        218.203.123.132  DNS        80 Standard query 0x147b A oth.eve.mdt.qq.com
92097 2020-03-23 14:14:48.407344 218.203.123.132      10.160.151.84    DNS       187 Standard query response 0x147b A oth.eve.mdt.qq.com A 121.51.158.17 A 121.51.18.236 A 121.51.158.120 A …
92098 2020-03-23 14:14:48.407623 10.160.151.84        218.203.123.132  ICMP      215 Destination unreachable (Port unreachable)
92894 2020-03-23 14:15:07.971458 218.203.123.132      10.160.151.84    DNS       187 Standard query response 0x147b A oth.eve.mdt.qq.com A 121.51.158.18 A 121.51.18.229 A 121.51.18.236 A 1…
92895 2020-03-23 14:15:07.971836 10.160.151.84        218.203.123.132  ICMP      215 Destination unreachable (Port unreachable)
```

过滤特定URL DNS的方法 **dns.qry.name contains “XXX”**， 譬如过滤baidu DNS查询包

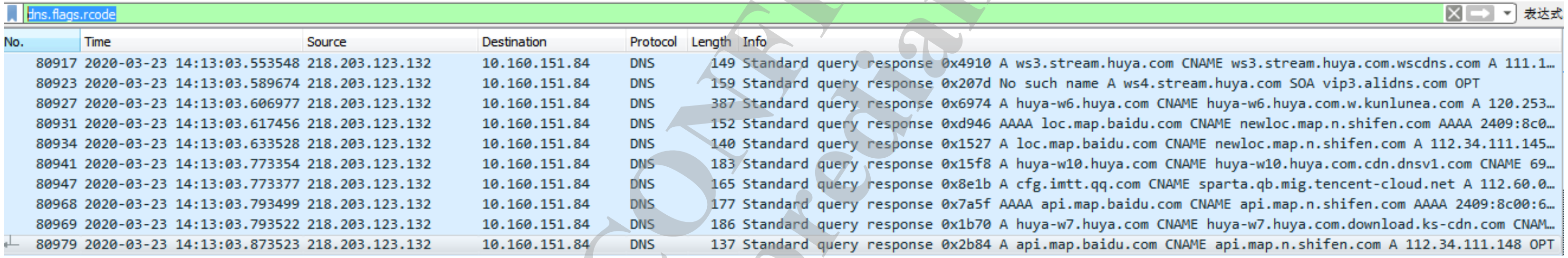


```
dns.qry.name contains "baidu"                                                                                    表达式

No.      Time                       Source            Destination      Protocol Length Info
   80928 2020-03-23 14:13:03.607371 10.160.151.84     218.203.123.132  DNS          79 Standard query 0xd946 AAAA loc.map.baidu.com
   80931 2020-03-23 14:13:03.617456 218.203.123.132   10.160.151.84    DNS         152 Standard query response 0xd946 AAAA loc.map.baidu.com CNAME newloc.map.n.shifen.com AAAA 2409:8c0…
   80932 2020-03-23 14:13:03.617983 10.160.151.84     218.203.123.132  DNS          79 Standard query 0x1527 A loc.map.baidu.com
   80934 2020-03-23 14:13:03.633528 218.203.123.132   10.160.151.84    DNS         140 Standard query response 0x1527 A loc.map.baidu.com CNAME newloc.map.n.shifen.com A 112.34.111.145…
   80962 2020-03-23 14:13:03.775393 10.160.151.84     218.203.123.132  DNS          79 Standard query 0x7a5f AAAA api.map.baidu.com
   80968 2020-03-23 14:13:03.793499 218.203.123.132   10.160.151.84    DNS         177 Standard query response 0x7a5f AAAA api.map.baidu.com CNAME api.map.n.shifen.com AAAA 2409:8c00:6…
   80970 2020-03-23 14:13:03.794070 10.160.151.84     218.203.123.132  DNS          79 Standard query 0x2b84 A api.map.baidu.com
   80979 2020-03-23 14:13:03.873523 218.203.123.132   10.160.151.84    DNS         137 Standard query response 0x2b84 A api.map.baidu.com CNAME api.map.n.shifen.com A 112.34.111.148 OPT
   1012… 2020-03-23 14:17:52.235997 10.160.151.84     218.203.123.132  DNS          73 Standard query 0xe254 AAAA m.baidu.com
   1013… 2020-03-23 14:17:53.148185 2409:897a:40:6f64:a… 2409:807a:2001::1 DNS       93 Standard query 0x5444 A m.baidu.com
```

Third app network access issue

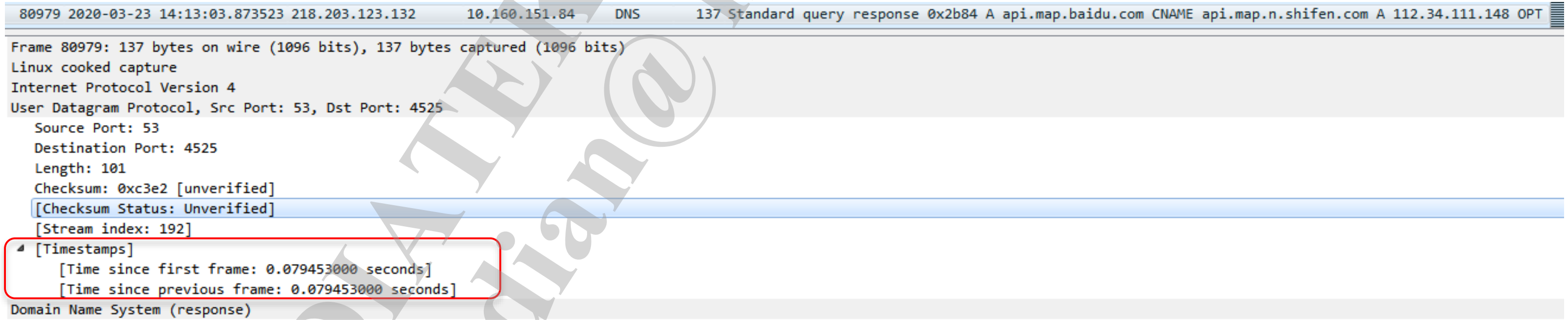
– Case 1 DNS查询延时

过滤DNS 响应数据包 **dns.flags.rcode**



```
No.   Time                        Source          Destination    Protocol Length Info
80917 2020-03-23 14:13:03.553548 218.203.123.132 10.160.151.84  DNS      149 Standard query response 0x4910 A ws3.stream.huya.com CNAME ws3.stream.huya.com.wscdns.com A 111.1…
80923 2020-03-23 14:13:03.589674 218.203.123.132 10.160.151.84  DNS      159 Standard query response 0x207d No such name A ws4.stream.huya.com SOA vip3.alidns.com OPT
80927 2020-03-23 14:13:03.606977 218.203.123.132 10.160.151.84  DNS      387 Standard query response 0x6974 A huya-w6.huya.com CNAME huya-w6.huya.com.w.kunlunea.com A 120.253…
80931 2020-03-23 14:13:03.617456 218.203.123.132 10.160.151.84  DNS      152 Standard query response 0xd946 AAAA loc.map.baidu.com CNAME newloc.map.n.shifen.com AAAA 2409:8c0…
80934 2020-03-23 14:13:03.633528 218.203.123.132 10.160.151.84  DNS      140 Standard query response 0x1527 A loc.map.baidu.com CNAME newloc.map.n.shifen.com A 112.34.111.145…
80941 2020-03-23 14:13:03.773354 218.203.123.132 10.160.151.84  DNS      183 Standard query response 0x15f8 A huya-w10.huya.com CNAME huya-w10.huya.com.cdn.dnsv1.com CNAME 69…
80947 2020-03-23 14:13:03.773377 218.203.123.132 10.160.151.84  DNS      165 Standard query response 0x8e1b A cfg.imtt.qq.com CNAME sparta.qb.mig.tencent-cloud.net A 112.60.0…
80968 2020-03-23 14:13:03.793499 218.203.123.132 10.160.151.84  DNS      177 Standard query response 0x7a5f AAAA api.map.baidu.com CNAME api.map.n.shifen.com AAAA 2409:8c00:6…
80969 2020-03-23 14:13:03.793522 218.203.123.132 10.160.151.84  DNS      186 Standard query response 0x1b70 A huya-w7.huya.com CNAME huya-w7.huya.com.download.ks-cdn.com CNAM…
80979 2020-03-23 14:13:03.873523 218.203.123.132 10.160.151.84  DNS      137 Standard query response 0x2b84 A api.map.baidu.com CNAME api.map.n.shifen.com A 112.34.111.148 OPT
```

那么如何判断有多少DNS 查询包有延时呢？这个时候就需要用到TCP和UDP包的一个选项:timestamps，过滤时可使用**udp.time_relative**(详细解释见下一个case)

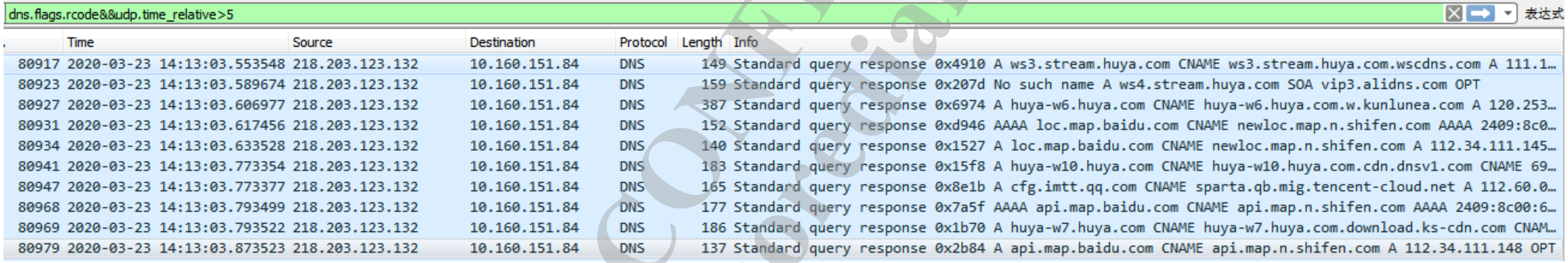


```
80979 2020-03-23 14:13:03.873523 218.203.123.132    10.160.151.84    DNS    137 Standard query response 0x2b84 A api.map.baidu.com CNAME api.map.n.shifen.com A 112.34.111.148 OPT

Frame 80979: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits)
Linux cooked capture
Internet Protocol Version 4
User Datagram Protocol, Src Port: 53, Dst Port: 4525
    Source Port: 53
    Destination Port: 4525
    Length: 101
    Checksum: 0xc3e2 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 192]
  ◢ [Timestamps]
        [Time since first frame: 0.079453000 seconds]
        [Time since previous frame: 0.079453000 seconds]
Domain Name System (response)
```

- # Third app network access issue

  - ## Case 1 DNS查询延时

    结合上述过滤条件，使用dns.flags.rcode&&udp.time_relative>5可以过滤出超时响应的DNS查询



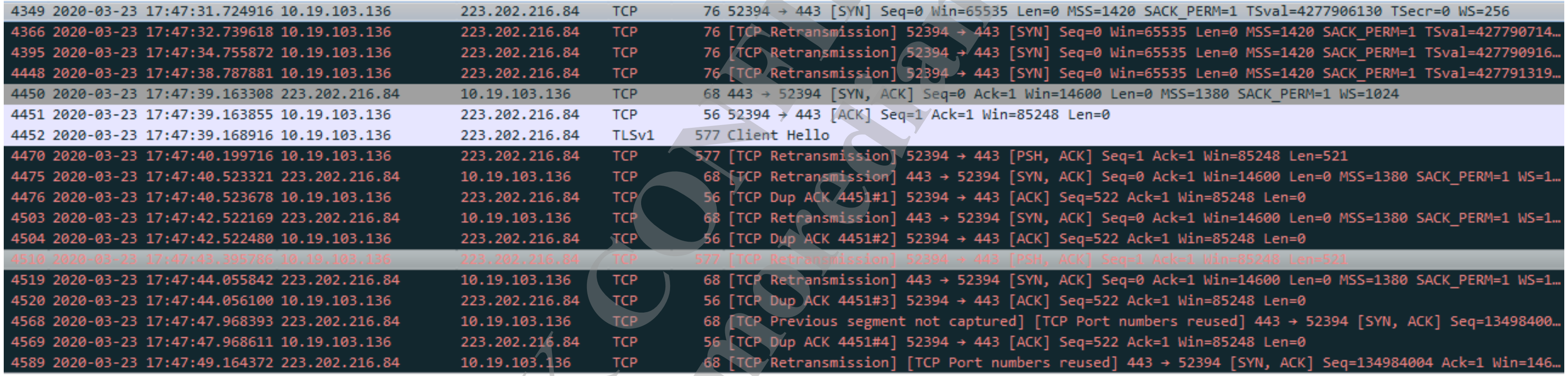
```
dns.flags.rcode&&udp.time_relative>5                                                                  表达式

      Time                       Source            Destination     Protocol Length Info
80917 2020-03-23 14:13:03.553548 218.203.123.132   10.160.151.84   DNS      149 Standard query response 0x4910 A ws3.stream.huya.com CNAME ws3.stream.huya.com.wscdns.com A 111.1…
80923 2020-03-23 14:13:03.589674 218.203.123.132   10.160.151.84   DNS      159 Standard query response 0x207d No such name A ws4.stream.huya.com SOA vip3.alidns.com OPT
80927 2020-03-23 14:13:03.606977 218.203.123.132   10.160.151.84   DNS      387 Standard query response 0x6974 A huya-w6.huya.com CNAME huya-w6.huya.com.w.kunlunea.com A 120.253…
80931 2020-03-23 14:13:03.617456 218.203.123.132   10.160.151.84   DNS      152 Standard query response 0xd946 AAAA loc.map.baidu.com CNAME newloc.map.n.shifen.com AAAA 2409:8c0…
80934 2020-03-23 14:13:03.633528 218.203.123.132   10.160.151.84   DNS      140 Standard query response 0x1527 A loc.map.baidu.com CNAME newloc.map.n.shifen.com A 112.34.111.145…
80941 2020-03-23 14:13:03.773354 218.203.123.132   10.160.151.84   DNS      183 Standard query response 0x15f8 A huya-w10.huya.com CNAME huya-w10.huya.com.cdn.dnsv1.com CNAME 69…
80947 2020-03-23 14:13:03.773377 218.203.123.132   10.160.151.84   DNS      165 Standard query response 0x8e1b A cfg.imtt.qq.com CNAME sparta.qb.mig.tencent-cloud.net A 112.60.0…
80968 2020-03-23 14:13:03.793499 218.203.123.132   10.160.151.84   DNS      177 Standard query response 0x7a5f AAAA api.map.baidu.com CNAME api.map.n.shifen.com AAAA 2409:8c00:6…
80969 2020-03-23 14:13:03.793522 218.203.123.132   10.160.151.84   DNS      186 Standard query response 0x1b70 A huya-w7.huya.com CNAME huya-w7.huya.com.download.ks-cdn.com CNAM…
80979 2020-03-23 14:13:03.873523 218.203.123.132   10.160.151.84   DNS      137 Standard query response 0x2b84 A api.map.baidu.com CNAME api.map.n.shifen.com A 112.34.111.148 OPT
```

如果想过滤出某一个URL DNS查询是否超时，还可以在上一步的过滤条件再&& dns.qry.name contains “XXX”. 合理使用过滤条件，可以帮助我们更好的从宏观上发现问题

Third app network access issue

– Case 2 TCP响应延时

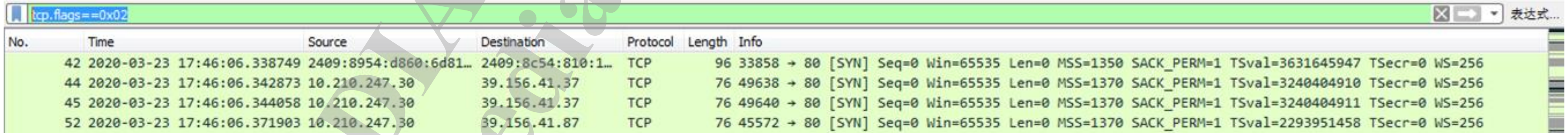
现象如下所示，从中可以看到TCP 三次握手就花了8秒， 而这8秒主要是syn ack 回复慢引起的。



```
4349 2020-03-23 17:47:31.724916 10.19.103.136     223.202.216.84    TCP      76 52394 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=4277906130 TSecr=0 WS=256
4366 2020-03-23 17:47:32.739618 10.19.103.136     223.202.216.84    TCP      76 [TCP Retransmission] 52394 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427790714…
4395 2020-03-23 17:47:34.755872 10.19.103.136     223.202.216.84    TCP      76 [TCP Retransmission] 52394 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427790916…
4448 2020-03-23 17:47:38.787881 10.19.103.136     223.202.216.84    TCP      76 [TCP Retransmission] 52394 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427791319…
4450 2020-03-23 17:47:39.163308 223.202.216.84    10.19.103.136     TCP      68 443 → 52394 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
4451 2020-03-23 17:47:39.163855 10.19.103.136     223.202.216.84    TCP      56 52394 → 443 [ACK] Seq=1 Ack=1 Win=85248 Len=0
4452 2020-03-23 17:47:39.168916 10.19.103.136     223.202.216.84    TLSv1   577 Client Hello
4470 2020-03-23 17:47:40.199716 10.19.103.136     223.202.216.84    TCP     577 [TCP Retransmission] 52394 → 443 [PSH, ACK] Seq=1 Ack=1 Win=85248 Len=521
4475 2020-03-23 17:47:40.523321 223.202.216.84    10.19.103.136     TCP      68 [TCP Retransmission] 443 → 52394 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1…
4476 2020-03-23 17:47:40.523678 10.19.103.136     223.202.216.84    TCP      56 [TCP Dup ACK 4451#1] 52394 → 443 [ACK] Seq=522 Ack=1 Win=85248 Len=0
4503 2020-03-23 17:47:42.522169 223.202.216.84    10.19.103.136     TCP      68 [TCP Retransmission] 443 → 52394 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1…
4504 2020-03-23 17:47:42.522480 10.19.103.136     223.202.216.84    TCP      56 [TCP Dup ACK 4451#2] 52394 → 443 [ACK] Seq=522 Ack=1 Win=85248 Len=0
4510 2020-03-23 17:47:43.395786 10.19.103.136     223.202.216.84    TCP     577 [TCP Retransmission] 52394 → 443 [PSH, ACK] Seq=1 Ack=1 Win=85248 Len=521
4519 2020-03-23 17:47:44.055842 223.202.216.84    10.19.103.136     TCP      68 [TCP Retransmission] 443 → 52394 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1…
4520 2020-03-23 17:47:44.056100 10.19.103.136     223.202.216.84    TCP      56 [TCP Dup ACK 4451#3] 52394 → 443 [ACK] Seq=522 Ack=1 Win=85248 Len=0
4568 2020-03-23 17:47:47.968393 223.202.216.84    10.19.103.136     TCP      68 [TCP Previous segment not captured] [TCP Port numbers reused] 443 → 52394 [SYN, ACK] Seq=13498400…
4569 2020-03-23 17:47:47.968611 10.19.103.136     223.202.216.84    TCP      56 [TCP Dup ACK 4451#4] 52394 → 443 [ACK] Seq=522 Ack=1 Win=85248 Len=0
4589 2020-03-23 17:47:49.164372 223.202.216.84    10.19.103.136     TCP      68 [TCP Retransmission] [TCP Port numbers reused] 443 → 52394 [SYN, ACK] Seq=134984004 Ack=1 Win=146…
```

但这只是其中某一条tcp trace ,不能管中窥豹，如果需要评估某一段时间网络的性能情况，则需要有一个整体比较。这个时候就需要掌握更多的wireshark使用技巧

过滤syn数据包 **tcp.flags==0x02**

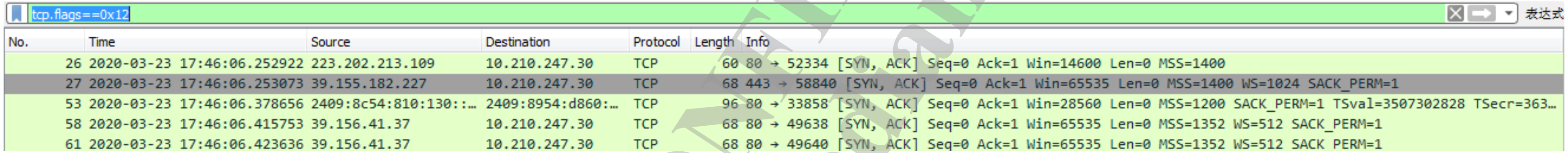


```
No.        Time                       Source                 Destination          Protocol  Length  Info
        42 2020-03-23 17:46:06.338749 2409:8954:d860:6d81…   2409:8c54:810:1…     TCP           96 33858 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1350 SACK_PERM=1 TSval=3631645947 TSecr=0 WS=256
        44 2020-03-23 17:46:06.342873 10.210.247.30          39.156.41.37         TCP           76 49638 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1370 SACK_PERM=1 TSval=3240404910 TSecr=0 WS=256
        45 2020-03-23 17:46:06.344058 10.210.247.30          39.156.41.37         TCP           76 49640 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1370 SACK_PERM=1 TSval=3240404911 TSecr=0 WS=256
        52 2020-03-23 17:46:06.371903 10.210.247.30          39.156.41.87         TCP           76 45572 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1370 SACK_PERM=1 TSval=2293951458 TSecr=0 WS=256
```

Third app network access issue

Case 2 TCP响应延时

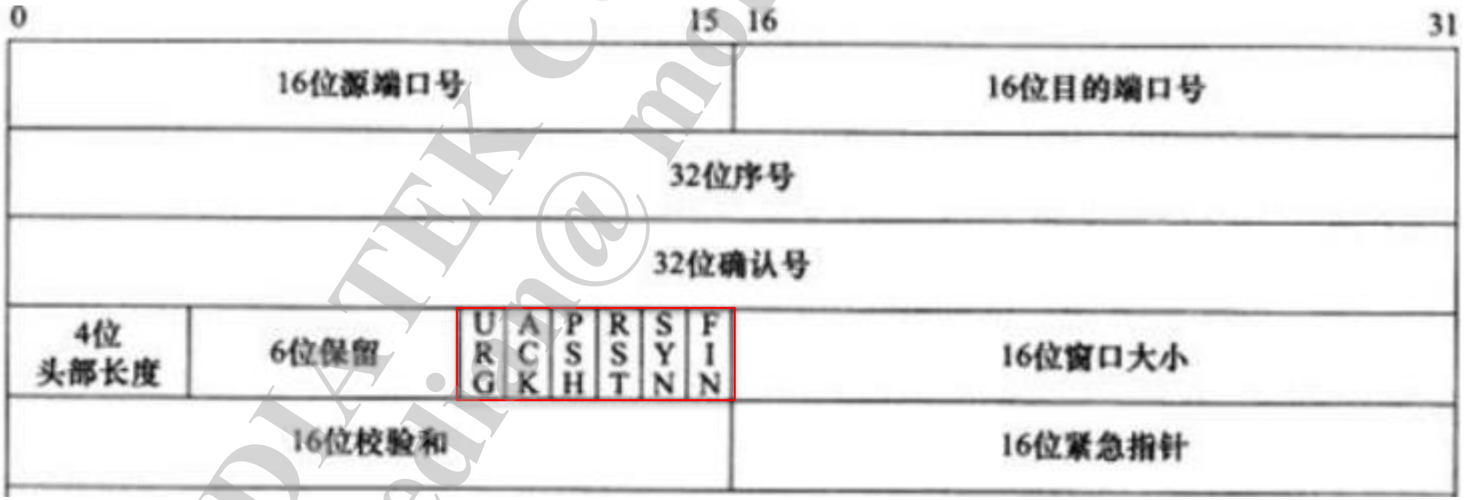
过滤syn ack数据包 **tcp.flags==0x12**



tcp.flags==0x12

No.	Time	Source	Destination	Protocol	Length	Info
26	2020-03-23 17:46:06.252922	223.202.213.109	10.210.247.30	TCP	60	80 → 52334 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1400
27	2020-03-23 17:46:06.253073	39.155.182.227	10.210.247.30	TCP	68	443 → 58840 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 WS=1024 SACK_PERM=1
53	2020-03-23 17:46:06.378656	2409:8c54:810:130::…	2409:8954:d860:…	TCP	96	80 → 33858 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1200 SACK_PERM=1 TSval=3507302828 TSecr=363…
58	2020-03-23 17:46:06.415753	39.156.41.37	10.210.247.30	TCP	68	80 → 49638 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1352 WS=512 SACK_PERM=1
61	2020-03-23 17:46:06.423636	39.156.41.37	10.210.247.30	TCP	68	80 → 49640 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1352 WS=512 SACK PERM=1

为什么这样能过滤出想要的数据包？FIN包,RST 包过滤条件如何写？ ？ ？



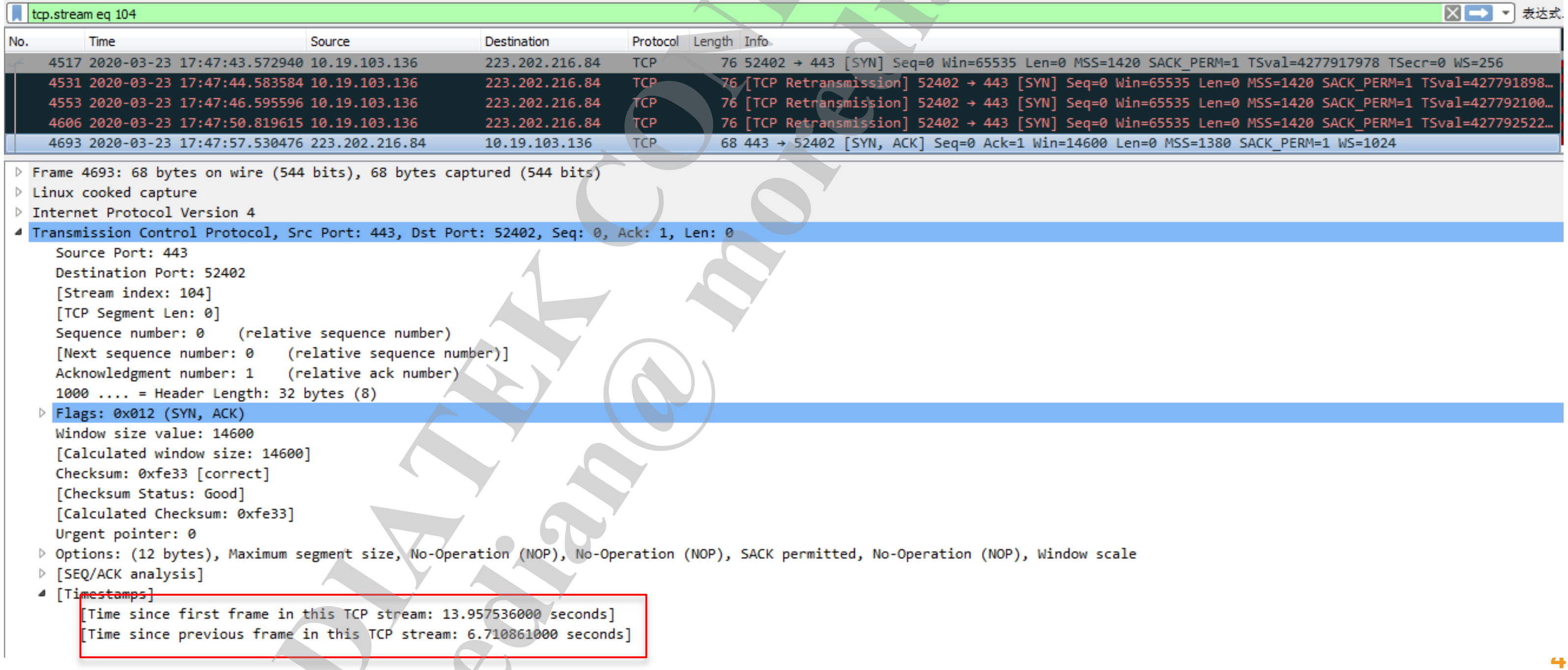
Third app network access issue

– Case 2 TCP响应延时

如何查看TCP 三次握手时syn ack 的响应时间？ **Timestamps , 其中**

tcp.time_relative 是time since first frame in this tcp stream

tcp.time_delta 是time since previous frame in this tcp stream



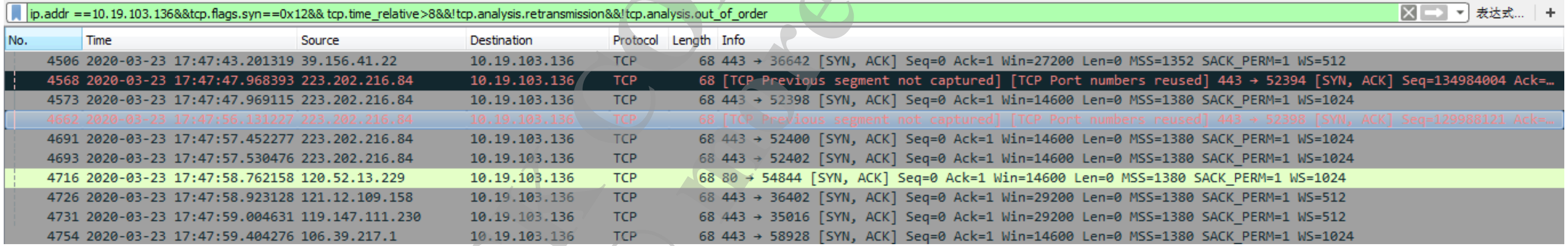
```
tcp.stream eq 104

No.   Time                       Source         Destination    Protocol Length Info
 4517 2020-03-23 17:47:43.572940 10.19.103.136  223.202.216.84 TCP       76 52402 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=4277917978 TSecr=0 WS=256
 4531 2020-03-23 17:47:44.583584 10.19.103.136  223.202.216.84 TCP       76 [TCP Retransmission] 52402 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427791898…
 4553 2020-03-23 17:47:46.595596 10.19.103.136  223.202.216.84 TCP       76 [TCP Retransmission] 52402 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427792100…
 4606 2020-03-23 17:47:50.819615 10.19.103.136  223.202.216.84 TCP       76 [TCP Retransmission] 52402 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1420 SACK_PERM=1 TSval=427792522…
 4693 2020-03-23 17:47:57.530476 223.202.216.84 10.19.103.136  TCP       68 443 → 52402 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
```

```
▷ Frame 4693: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
▷ Linux cooked capture
▷ Internet Protocol Version 4
◢ Transmission Control Protocol, Src Port: 443, Dst Port: 52402, Seq: 0, Ack: 1, Len: 0
     Source Port: 443
     Destination Port: 52402
     [Stream index: 104]
     [TCP Segment Len: 0]
     Sequence number: 0    (relative sequence number)
     [Next sequence number: 0    (relative sequence number)]
     Acknowledgment number: 1    (relative ack number)
     1000 .... = Header Length: 32 bytes (8)
  ▷ Flags: 0x012 (SYN, ACK)
     Window size value: 14600
     [Calculated window size: 14600]
     Checksum: 0xfe33 [correct]
     [Checksum Status: Good]
     [Calculated Checksum: 0xfe33]
     Urgent pointer: 0
  ▷ Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  ▷ [SEQ/ACK analysis]
  ◢ [Timestamps]
       [Time since first frame in this TCP stream: 13.957536000 seconds]
       [Time since previous frame in this TCP stream: 6.710861000 seconds]
```

Third app network access issue

- Case 2 TCP响应延时

结合前面的分析，通过过滤条件”ip.addr ==10.19.103.136&&tcp.flags.syn==0x12&& tcp.time_relative>8”，我们就可以过滤出syn ack包响应超过8秒的数据包（为了避免重传和乱序的影响，可以在过滤条件中加入&&!tcp.analysis.retransmission&&!tcp.analysis.out_of_order）。另8秒在这里只是一个举例数字，实际分析中可以自己设置调整



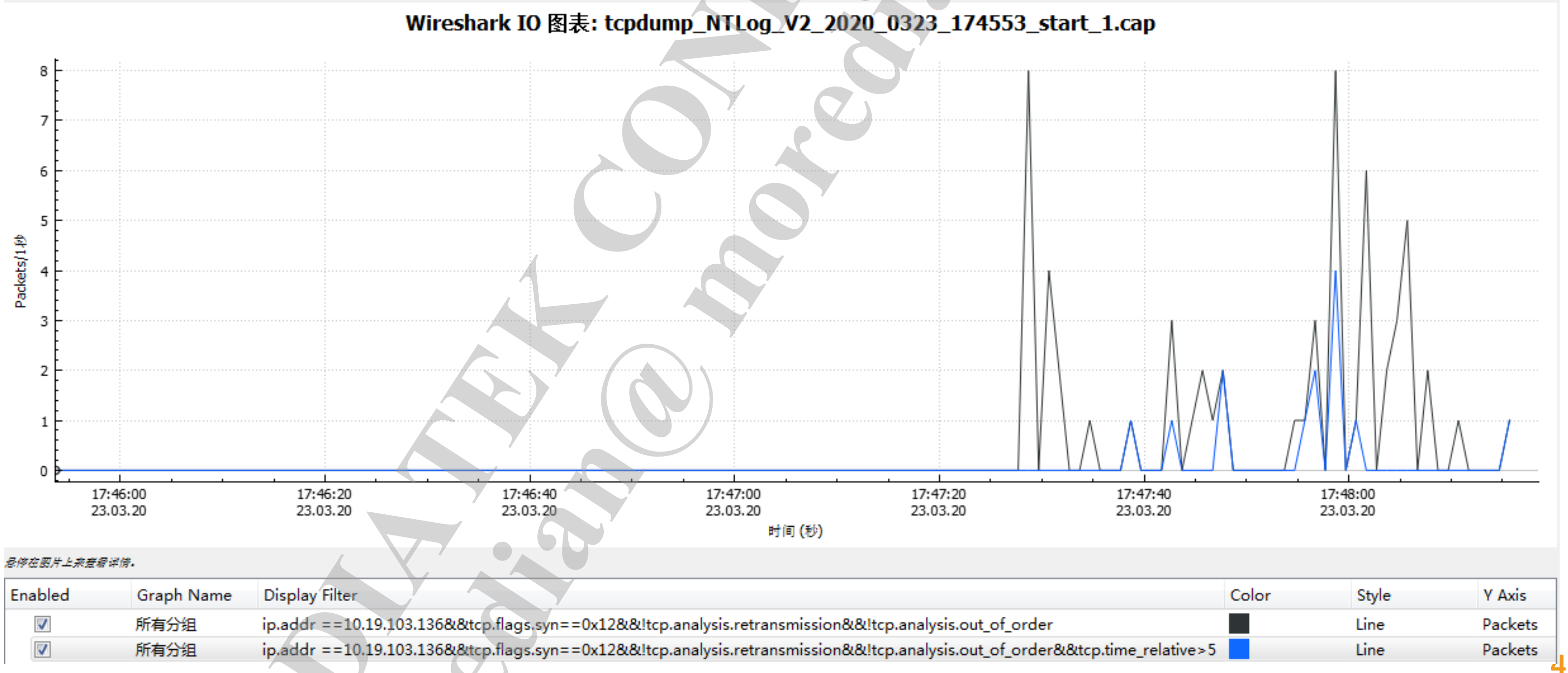
ip.addr ==10.19.103.136&&tcp.flags.syn==0x12&& tcp.time_relative>8&&!tcp.analysis.retransmission&&!tcp.analysis.out_of_order

No.	Time	Source	Destination	Protocol	Length	Info
4506	2020-03-23 17:47:43.201319	39.156.41.22	10.19.103.136	TCP	68	443 → 36642 [SYN, ACK] Seq=0 Ack=1 Win=27200 Len=0 MSS=1352 SACK_PERM=1 WS=512
4568	2020-03-23 17:47:47.968393	223.202.216.84	10.19.103.136	TCP	68	[TCP Previous segment not captured] [TCP Port numbers reused] 443 → 52394 [SYN, ACK] Seq=134984004 Ack=…
4573	2020-03-23 17:47:47.969115	223.202.216.84	10.19.103.136	TCP	68	443 → 52398 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
4662	2020-03-23 17:47:56.131227	223.202.216.84	10.19.103.136	TCP	68	[TCP Previous segment not captured] [TCP Port numbers reused] 443 → 52398 [SYN, ACK] Seq=129988121 Ack=…
4691	2020-03-23 17:47:57.452277	223.202.216.84	10.19.103.136	TCP	68	443 → 52400 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
4693	2020-03-23 17:47:57.530476	223.202.216.84	10.19.103.136	TCP	68	443 → 52402 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
4716	2020-03-23 17:47:58.762158	120.52.13.229	10.19.103.136	TCP	68	80 → 54844 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024
4726	2020-03-23 17:47:58.923128	121.12.109.158	10.19.103.136	TCP	68	443 → 36402 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=512
4731	2020-03-23 17:47:59.004631	119.147.111.230	10.19.103.136	TCP	68	443 → 35016 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=512
4754	2020-03-23 17:47:59.404276	106.39.217.1	10.19.103.136	TCP	68	443 → 58928 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=1024

Third app network access issue

Case 2 TCP响应延时

上图只是过滤出了syn ack 响应超出某一个时间点的包，如果想知道超时响应数据包占总数据包的比例，我们可以使用IO Grapha作更直观的比较



悬停在图片上来查看详情。

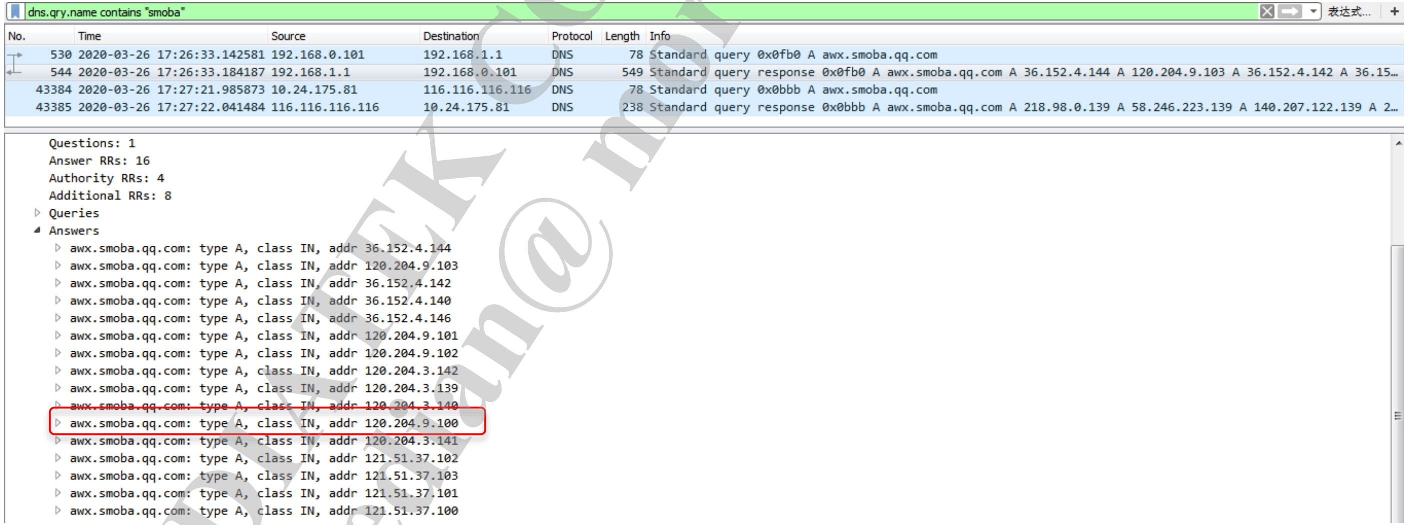
Enabled	Graph Name	Display Filter	Color	Style	Y Axis
☑	所有分组	ip.addr ==10.19.103.136&&tcp.flags.syn==0x12&&!tcp.analysis.retransmission&&!tcp.analysis.out_of_order	■	Line	Packets
☑	所有分组	ip.addr ==10.19.103.136&&tcp.flags.syn==0x12&&!tcp.analysis.retransmission&&!tcp.analysis.out_of_order&&tcp.time_relative>5	🟦	Line	Packets

Third app network access issue

– Case 3 王者荣耀卡顿问题分析

最近几年，手机的发展是越来越注重用户体验，而游戏卡顿是用户体验常见抱怨之一。游戏卡顿的实质是数据包交互有延时，因此，分析游戏卡顿最直接也是最困难的一个地方就是如何正确找出游戏交互的数据流。本节以王者荣耀为例进行分析。

游戏开局后，与手机进行实时交互的server 的URL是awx.smoba.qq.com（由于地域不同，URL可能略有差异，但一般都会带有smoba字符串），因此，首先可以通过dns.qry.name contains “smoba” 找出该URL的IP地址



```
No.    Time                       Source           Destination      Protocol Length Info
   530 2020-03-26 17:26:33.142581 192.168.0.101    192.168.1.1      DNS        78 Standard query 0x0fb0 A awx.smoba.qq.com
   544 2020-03-26 17:26:33.184187 192.168.1.1      192.168.0.101    DNS       549 Standard query response 0x0fb0 A awx.smoba.qq.com A 36.152.4.144 A 120.204.9.103 A 36.152.4.142 A 36.15…
 43384 2020-03-26 17:27:21.985873 10.24.175.81     116.116.116.116  DNS        78 Standard query 0x0bbb A awx.smoba.qq.com
 43385 2020-03-26 17:27:22.041484 116.116.116.116  10.24.175.81     DNS       238 Standard query response 0x0bbb A awx.smoba.qq.com A 218.98.0.139 A 58.246.223.139 A 140.207.122.139 A 2…
```

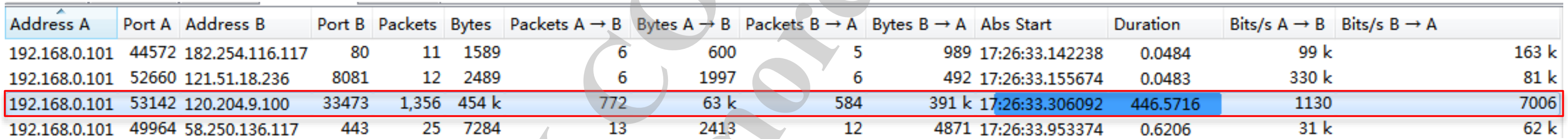
```
    Questions: 1
    Answer RRs: 16
    Authority RRs: 4
    Additional RRs: 8
  ▷ Queries
  ◢ Answers
    ▷ awx.smoba.qq.com: type A, class IN, addr 36.152.4.144
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.9.103
    ▷ awx.smoba.qq.com: type A, class IN, addr 36.152.4.142
    ▷ awx.smoba.qq.com: type A, class IN, addr 36.152.4.140
    ▷ awx.smoba.qq.com: type A, class IN, addr 36.152.4.146
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.9.101
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.9.102
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.3.142
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.3.139
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.3.140
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.9.100
    ▷ awx.smoba.qq.com: type A, class IN, addr 120.204.3.141
    ▷ awx.smoba.qq.com: type A, class IN, addr 121.51.37.102
    ▷ awx.smoba.qq.com: type A, class IN, addr 121.51.37.103
    ▷ awx.smoba.qq.com: type A, class IN, addr 121.51.37.101
    ▷ awx.smoba.qq.com: type A, class IN, addr 121.51.37.100
```

Third app network access issue

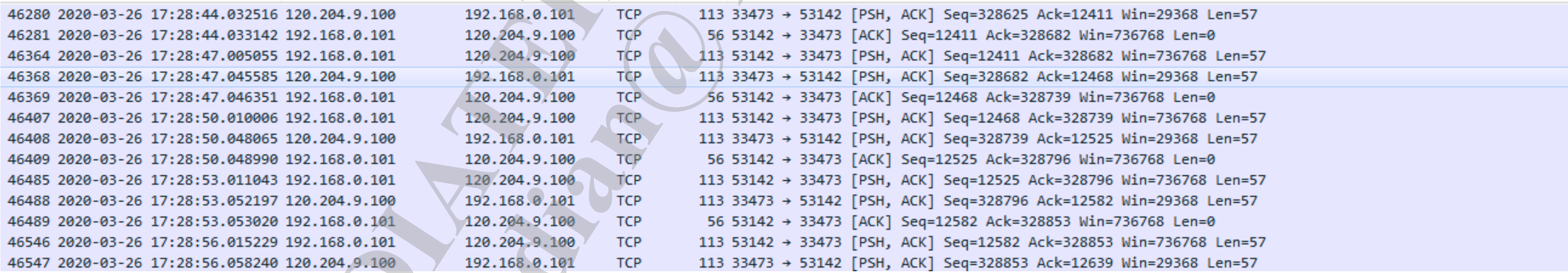
– Case 3 王者荣耀卡顿问题分析

游戏实时交互过程中，会建立一条TCP长连接数据流和两条UDP流（心跳包和游戏交互的实时数据）。正常情况下这三条流的目的地址都是smoba DNS 查询的IP地址中一个。在游戏过程中，与手机实时交互的server可能会有更新改变.

TCP 长连接，前期会有大量游戏数据通过这条流交互，后期差不多就会维持每3秒交互一次数据包，因此，通过查看这条流的RTT时延，也可以大致判断出游戏时延情况



Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Abs Start	Duration	Bits/s A → B	Bits/s B → A
192.168.0.101	44572	182.254.116.117	80	11	1589	6	600	5	989	17:26:33.142238	0.0484	99 k	163 k
192.168.0.101	52660	121.51.18.236	8081	12	2489	6	1997	6	492	17:26:33.155674	0.0483	330 k	81 k
192.168.0.101	53142	120.204.9.100	33473	1,356	454 k	772	63 k	584	391 k	17:26:33.306092	446.5716	1130	7006
192.168.0.101	49964	58.250.136.117	443	25	7284	13	2413	12	4871	17:26:33.953374	0.6206	31 k	62 k

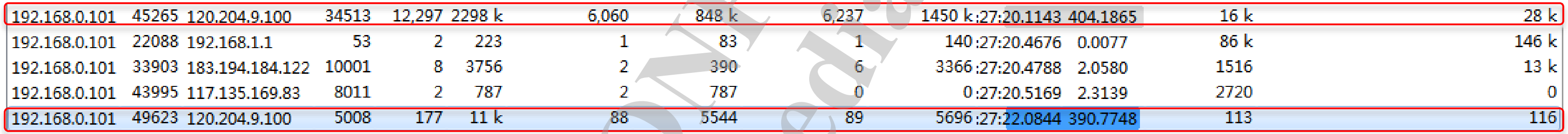


```
46280 2020-03-26 17:28:44.032516 120.204.9.100    192.168.0.101   TCP   113 33473 → 53142 [PSH, ACK] Seq=328625 Ack=12411 Win=29368 Len=57
46281 2020-03-26 17:28:44.033142 192.168.0.101    120.204.9.100   TCP    56 53142 → 33473 [ACK] Seq=12411 Ack=328682 Win=736768 Len=0
46364 2020-03-26 17:28:47.005055 192.168.0.101    120.204.9.100   TCP   113 53142 → 33473 [PSH, ACK] Seq=12411 Ack=328682 Win=736768 Len=57
46368 2020-03-26 17:28:47.045585 120.204.9.100    192.168.0.101   TCP   113 33473 → 53142 [PSH, ACK] Seq=328682 Ack=12468 Win=29368 Len=57
46369 2020-03-26 17:28:47.046351 192.168.0.101    120.204.9.100   TCP    56 53142 → 33473 [ACK] Seq=12468 Ack=328739 Win=736768 Len=0
46407 2020-03-26 17:28:50.010006 192.168.0.101    120.204.9.100   TCP   113 53142 → 33473 [PSH, ACK] Seq=12468 Ack=328739 Win=736768 Len=57
46408 2020-03-26 17:28:50.048065 120.204.9.100    192.168.0.101   TCP   113 33473 → 53142 [PSH, ACK] Seq=328739 Ack=12525 Win=29368 Len=57
46409 2020-03-26 17:28:50.048990 192.168.0.101    120.204.9.100   TCP    56 53142 → 33473 [ACK] Seq=12525 Ack=328796 Win=736768 Len=0
46485 2020-03-26 17:28:53.011043 192.168.0.101    120.204.9.100   TCP   113 53142 → 33473 [PSH, ACK] Seq=12525 Ack=328796 Win=736768 Len=57
46488 2020-03-26 17:28:53.052197 120.204.9.100    192.168.0.101   TCP   113 33473 → 53142 [PSH, ACK] Seq=328796 Ack=12582 Win=29368 Len=57
46489 2020-03-26 17:28:53.053020 192.168.0.101    120.204.9.100   TCP    56 53142 → 33473 [ACK] Seq=12582 Ack=328853 Win=736768 Len=0
46546 2020-03-26 17:28:56.015229 192.168.0.101    120.204.9.100   TCP   113 53142 → 33473 [PSH, ACK] Seq=12582 Ack=328853 Win=736768 Len=57
46547 2020-03-26 17:28:56.058240 120.204.9.100    192.168.0.101   TCP   113 33473 → 53142 [PSH, ACK] Seq=328853 Ack=12639 Win=29368 Len=57
```

Third app network access issue

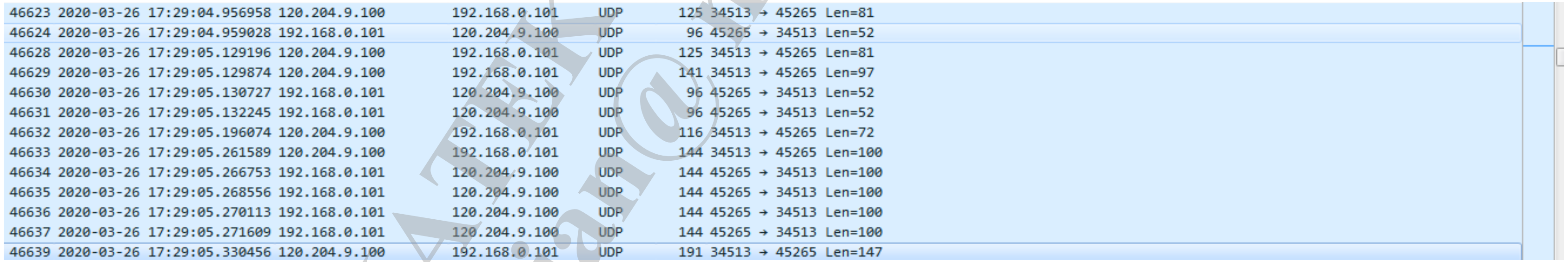
Case 3 王者荣耀卡顿问题分析

两条 UDP连接，一条负责游戏过程中数据的实时交互，另一条负责类似于心跳探测，每5秒发送一次。目的端口一般是**5008**



192.168.0.101	45265	120.204.9.100	34513	12,297	2298 k	6,060	848 k	6,237	1450 k	:27:20.1143	404.1865	16 k	28 k
192.168.0.101	22088	192.168.1.1	53	2	223	1	83	1	140	:27:20.4676	0.0077	86 k	146 k
192.168.0.101	33903	183.194.184.122	10001	8	3756	2	390	6	3366	:27:20.4788	2.0580	1516	13 k
192.168.0.101	43995	117.135.169.83	8011	2	787	2	787	0	0	:27:20.5169	2.3139	2720	0
192.168.0.101	49623	120.204.9.100	5008	177	11 k	88	5544	89	5696	:27:22.0844	390.7748	113	116

实时数据交互的UDP流，从图中可以看到，每条流的数据量不大，都是server数据包发送的间隔大概是60ms左右，当server发向手机端的数据包大量超过200~300 ms 以上，这时候就会产生卡顿

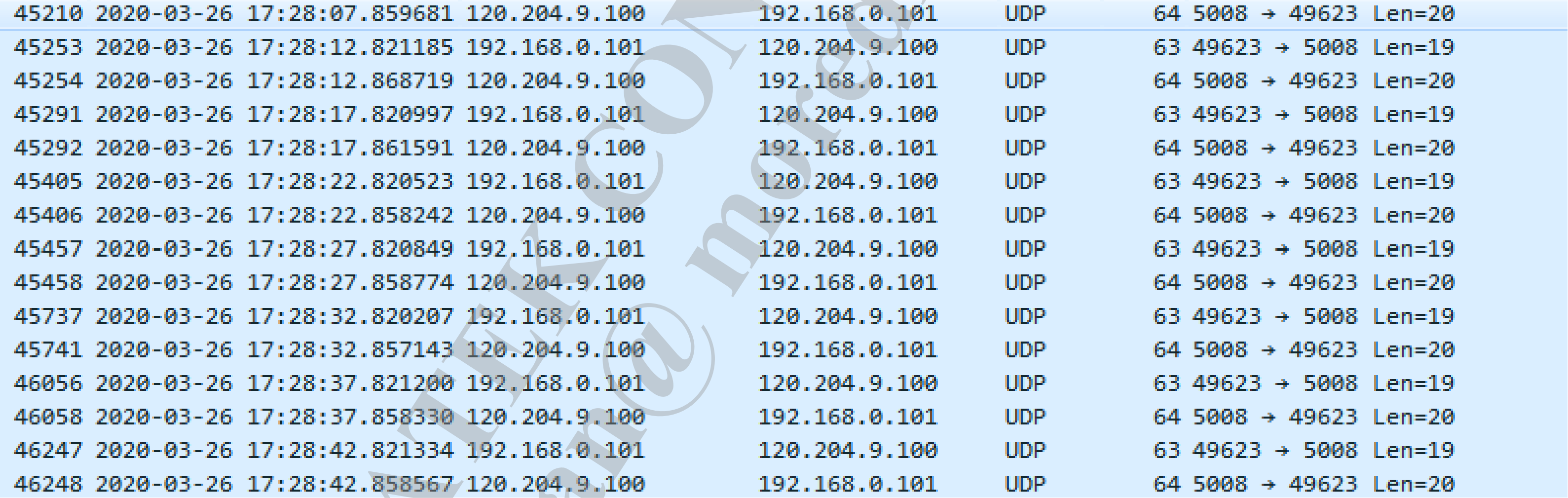


```
46623 2020-03-26 17:29:04.956958 120.204.9.100     192.168.0.101  UDP   125 34513 → 45265 Len=81
46624 2020-03-26 17:29:04.959028 192.168.0.101     120.204.9.100  UDP    96 45265 → 34513 Len=52
46628 2020-03-26 17:29:05.129196 120.204.9.100     192.168.0.101  UDP   125 34513 → 45265 Len=81
46629 2020-03-26 17:29:05.129874 120.204.9.100     192.168.0.101  UDP   141 34513 → 45265 Len=97
46630 2020-03-26 17:29:05.130727 192.168.0.101     120.204.9.100  UDP    96 45265 → 34513 Len=52
46631 2020-03-26 17:29:05.132245 192.168.0.101     120.204.9.100  UDP    96 45265 → 34513 Len=52
46632 2020-03-26 17:29:05.196074 120.204.9.100     192.168.0.101  UDP   116 34513 → 45265 Len=72
46633 2020-03-26 17:29:05.261589 120.204.9.100     192.168.0.101  UDP   144 34513 → 45265 Len=100
46634 2020-03-26 17:29:05.266753 192.168.0.101     120.204.9.100  UDP   144 45265 → 34513 Len=100
46635 2020-03-26 17:29:05.268556 192.168.0.101     120.204.9.100  UDP   144 45265 → 34513 Len=100
46636 2020-03-26 17:29:05.270113 192.168.0.101     120.204.9.100  UDP   144 45265 → 34513 Len=100
46637 2020-03-26 17:29:05.271609 192.168.0.101     120.204.9.100  UDP   144 45265 → 34513 Len=100
46639 2020-03-26 17:29:05.330456 120.204.9.100     192.168.0.101  UDP   191 34513 → 45265 Len=147
```

- ## Third app network access issue

  - ### Case 3 王者荣耀卡顿问题分析

数据量小，类似于心跳探测的UDP流如下所示，其发送固定大小的数据包，APP发19字节，服务器回20字节，每隔5s 发一次，server port=5008。有时客户提供的log没有包含实时server DNS查询过程，这时就可以通过udp.port ==5008 找到server 的IP地址



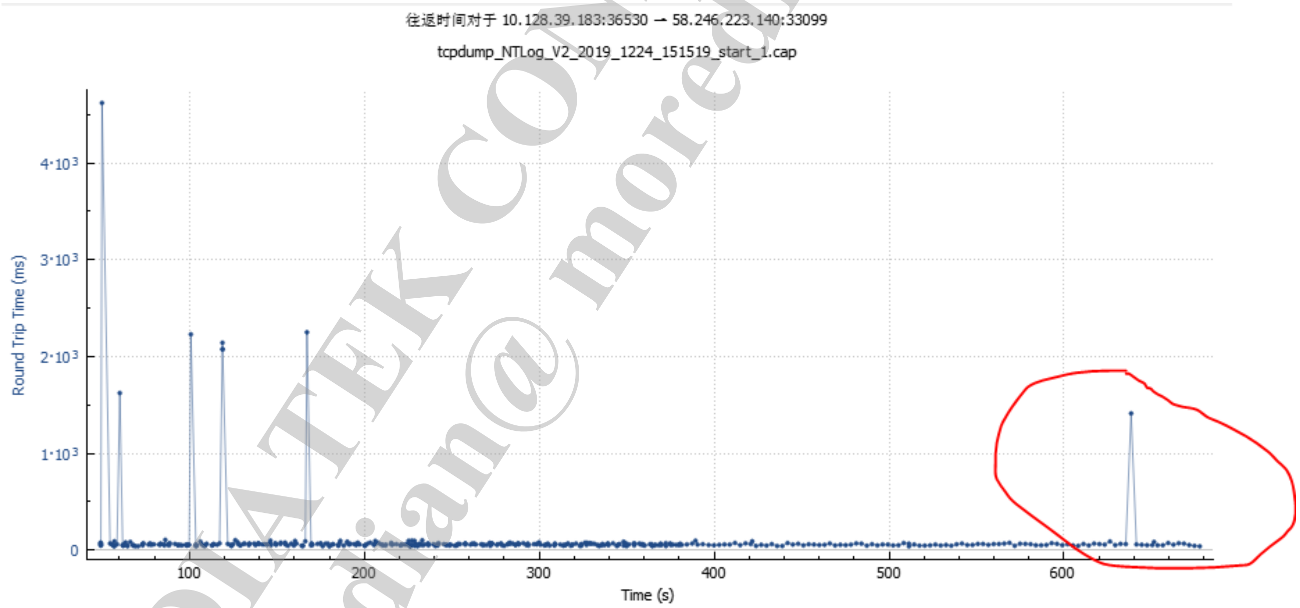
```
45210 2020-03-26 17:28:07.859681 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
45253 2020-03-26 17:28:12.821185 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
45254 2020-03-26 17:28:12.868719 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
45291 2020-03-26 17:28:17.820997 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
45292 2020-03-26 17:28:17.861591 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
45405 2020-03-26 17:28:22.820523 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
45406 2020-03-26 17:28:22.858242 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
45457 2020-03-26 17:28:27.820849 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
45458 2020-03-26 17:28:27.858774 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
45737 2020-03-26 17:28:32.820207 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
45741 2020-03-26 17:28:32.857143 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
46056 2020-03-26 17:28:37.821200 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
46058 2020-03-26 17:28:37.858330 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
46247 2020-03-26 17:28:42.821334 192.168.0.101       120.204.9.100     UDP         63 49623 → 5008 Len=19
46248 2020-03-26 17:28:42.858567 120.204.9.100       192.168.0.101     UDP         64 5008 → 49623 Len=20
```

Third app network access issue

– Case 3 王者荣耀卡顿问题分析

如何快速找出游戏卡顿的时间点？

一，是找到TCP长连接，通过IO Grapha画出其RTT时延图，找出其高时延点即可，如下所示

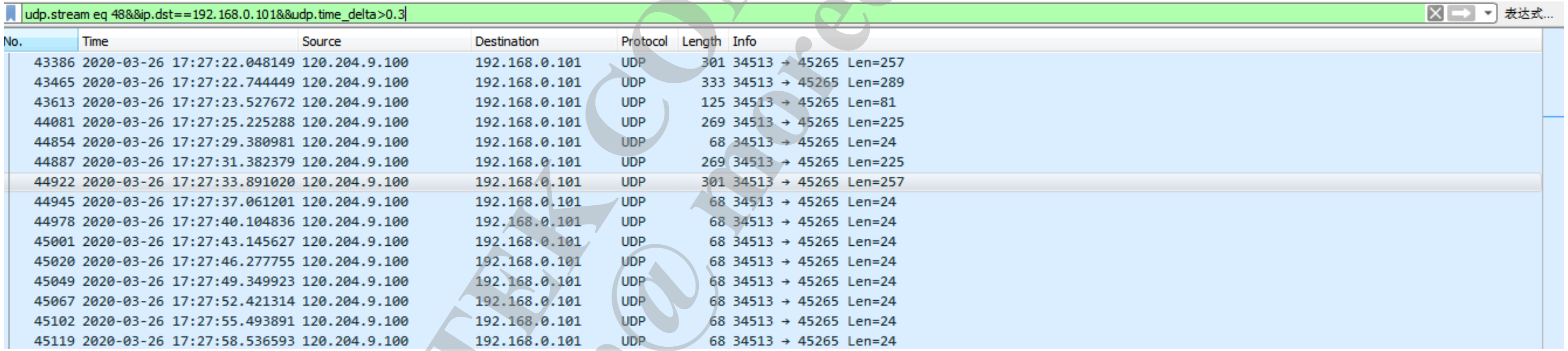


Third app network access issue

– Case 3 王者荣耀卡顿问题分析

如何快速找出游戏卡顿的时间点？

二，是通过UDP游戏实时交互这条流，过滤出server发向手机的UDP包时延超过200 ms以上的点，可以通过**udp.time_delta**来过滤。当然还可以观察另外一条UDP流，看其每5s 收发一次数据是否有中断时间段。



udp.stream eq 48&&ip.dst==192.168.0.101&&udp.time_delta>0.3

No.	Time	Source	Destination	Protocol	Length	Info
43386	2020-03-26 17:27:22.048149	120.204.9.100	192.168.0.101	UDP	301	34513 → 45265 Len=257
43465	2020-03-26 17:27:22.744449	120.204.9.100	192.168.0.101	UDP	333	34513 → 45265 Len=289
43613	2020-03-26 17:27:23.527672	120.204.9.100	192.168.0.101	UDP	125	34513 → 45265 Len=81
44081	2020-03-26 17:27:25.225288	120.204.9.100	192.168.0.101	UDP	269	34513 → 45265 Len=225
44854	2020-03-26 17:27:29.380981	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
44887	2020-03-26 17:27:31.382379	120.204.9.100	192.168.0.101	UDP	269	34513 → 45265 Len=225
44922	2020-03-26 17:27:33.891020	120.204.9.100	192.168.0.101	UDP	301	34513 → 45265 Len=257
44945	2020-03-26 17:27:37.061201	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
44978	2020-03-26 17:27:40.104836	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45001	2020-03-26 17:27:43.145627	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45020	2020-03-26 17:27:46.277755	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45049	2020-03-26 17:27:49.349923	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45067	2020-03-26 17:27:52.421314	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45102	2020-03-26 17:27:55.493891	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24
45119	2020-03-26 17:27:58.536593	120.204.9.100	192.168.0.101	UDP	68	34513 → 45265 Len=24

Thanks