



产品名称Product name	密级Confidentiality level
LTE	内部公开
产品版本Product version	Total 47pages 共47页

LTE数传问题分析Wireshark使用指导书

(仅供内部使用)
For internal use only

拟制: Prepared by	魏志 00128876		日期: Date	2011-12-14
审核: Reviewed by			日期: Date	
审核: Reviewed by			日期: Date	
批准: Granted by			日期: Date	



华为技术有限公司

Huawei Technologies Co., Ltd.

版权所有 侵权必究
All rights reserved

修订记录Revision record

日期 Date	修订版本 Revision version	修改描述 change Description	作者 Author
2011-12-14	1.0	初稿建立	魏志 00128876

目 录Table of Contents

1	概述	5
2	软件安装	5
2.1	安装WinPcap提示冲突且不可删除	5
2.2	安装Wireshark后网卡选项中找不到网卡	6
2.3	Windows 7下提示NPF驱动未运行，找不到网卡	6
3	数据采集	6
3.1	使用Wireshark采集数据	6
3.1.1	Wireshark抓包选项	8
3.1.2	捕捉滤波器	10
3.2	使用其它软件采集数据	11
3.2.1	Windows 7下Wireshark抓包找不到Huawei E398虚拟网卡，利用MS Network Monitor采集数据	11
4	软件设置	12
4.1	Wireshark基本界面	12
4.2	参数设置	13
4.2.1	如何配置Packet list pane（包列表显示区）显示项	13
4.2.2	IP协议配置	15
4.2.3	TCP协议配置	15
4.2.4	ESP协议配置	16
5	数据处理	16
5.1	常规技能	16
5.1.1	时间设置	16
5.1.2	如何快速找到某个包	17
5.1.3	指定协议解析	17
5.1.4	如何自动重组分片包	18
5.2	文件处理	18
5.2.1	仅保存需要的数据	18
5.2.2	多文件合并	20
5.2.3	转换文件格式输出	21
5.3	Wireshark自带的命令行工具	22
5.3.1	如何使用Wireshark命令行工具	22
5.3.2	利用editcap分割文件	23
6	数据分析	24
6.1	Wireshark数据分析基本知识	24
6.1.1	Wireshark基本标识	24
6.1.2	抓包统计 Summary	25
6.1.3	流量统计 IO Graphs	25
6.1.4	汇总所有日志的专家信息 Expert Info Composite	28
6.1.5	交互流图 Flow Graph	29
6.2	TCP基本参数分析	30
6.2.1	TCP接收窗口	30
6.2.2	TCP发送窗口	30
6.2.3	其它TCP基本参数	31
6.2.4	RTT环回时延	32
6.2.5	如何过滤某一条TCP链路	32
6.3	显示滤波器分析法	33

6.3.1	显示滤波器表达式.....	33
6.3.2	如何把抓包中的某个字段设为滤波器.....	34
6.3.3	常用显示滤波器.....	35
6.3.4	TCP显示滤波器族	35
6.3.5	利用显示滤波器分析丢包.....	37
6.3.6	利用显示滤波器分析乱序.....	37
6.3.7	利用显示滤波器分析窗口收缩.....	38
6.4	图形分析法.....	39
6.4.1	Wireshark的图形控制	39
6.4.2	TCP流量图 Throughput Graph.....	40
6.4.3	TCP接收窗口图 Window Scaling Graph.....	41
6.4.4	时序图介绍 Time-Sequence Graph.....	42
6.4.5	Stevens时序图 Time-Sequence Graph (Stevens)	43
6.4.6	tcptrace时序图Time-Sequence Graph (tcptrace).....	43
6.4.7	利用tcptrace时序图分析TCP链路异常.....	44
7	使用Wireshark分析LTE数传问题流程	45
7.1.1	常规步骤.....	45
7.1.2	使用Wireshark分析LTE数传问题流程图.....	47

1 概述

Wireshark是一款自由、开源的包分析软件。主要用于网络问题定位、分析，软件和通信协议开发及教育。Wireshark原名Ethereal，2006年5月因为商标问题改名Wireshark。

Wireshark是跨平台软件，可以运行在Microsoft Windows以及各类Unix衍生的操作系统下，在诸如Linux、Mac OS X、BSD、Solaris等各类操作系统下都可以见到它的身影。

Wireshark通过集成pcap（Packet CAPture）实现抓包。

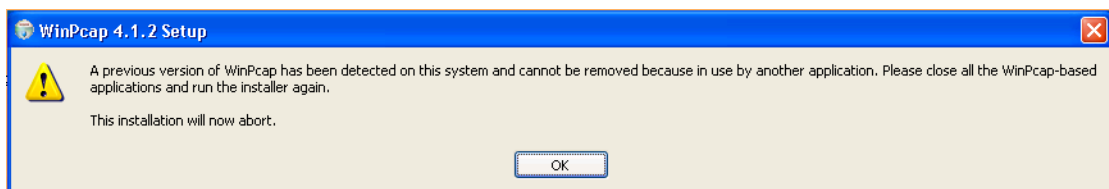
本文基于Wireshark 1.6.2版本写作，部分功能选单在之前的版本中并不存在，请注意保持版本同步。

如未特殊说明，以下描述均基于Windows平台。

2 软件安装

由于Wireshark通过pcap实现抓包，新版的Wireshark安装包集成了WinPcap的安装包。Wireshark和WinPcap的安装比较简单，这里就不赘述了，但要注意安装操作系统对应的版本（32-bit or 64-bit）。下面简单描述一下在安装过程中经常遇到的几个典型问题：

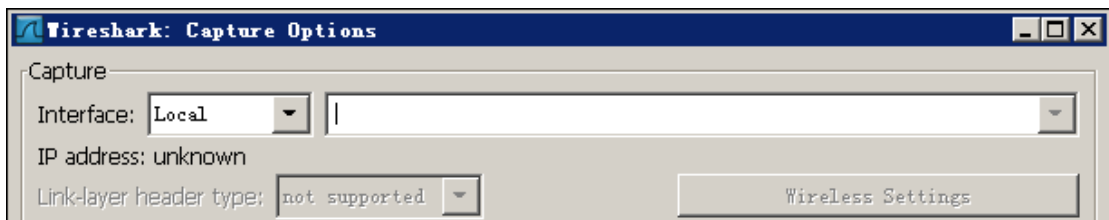
2.1 安装 WinPcap 提示冲突且不可删除



如图。由于WinPcap被广泛的应用于各类抓包软件，所以很可能你的机器上已经安装了其它版本的WinPcap，建议是按照提示卸载其它版本的WinPcap，安装Wireshark自带的版本。

由于我司某著名软件也使用WinPcap，对于我司的机器，可以不用安装新的WinPcap，一般情况下也不影响Wireshark的使用。

2.2 安装 Wireshark 后网卡选项中找不到网卡



如图，安装Wireshark后网卡选项中找不到本机的网卡实例，可以通过下面的步骤解决：

1. 重启电脑，如果重启后仍然不能解决，去第二步；
2. 卸载 Wireshark 和 WinPcap，删除如下文件后再重新安装 Wireshark 和 WinPcap：
WinPcap：
%windir%\system32\wpcap.dll， %windir%\system32\packet.dll
和 %windir%\system32\drivers\npf.sys， %windir%就是你的系统文件夹，比如说你的系统装在 C 盘，那么 %windir%\system32\wpcap.dll 就是 c:\windows\system32\wpcap.dll；

NOTE

如果删除以上文件时提示无法删除，可以选择重启机器到安全模式下再去删除或者安装 *unlocker* 软件解决。安装完 *unlocker* 后，右键点击需要删除的文件，选择 *Unlocker* 进行删除。

2.3 Windows 7 下提示 NPF 驱动未运行，找不到网卡



如图，这是由于WinPcap在win7中注册为非即插即用驱动程序（WinPcap Packet Driver (NPF)）引起的，需要以管理员身份手动启动NPF。程序 – 运行 – cmd，右键，以管理员身份运行，输入net start npf手动启动NPF服务即可。也可以输入sc config npf start=auto，以后每次系统启动将自动启动NPF服务。

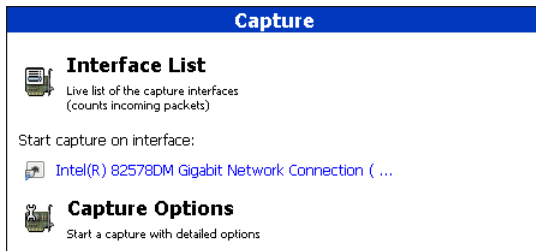
3 数据采集

3.1 使用 Wireshark 采集数据

Wireshark功能相当强大，以下重点描述LTE数传问题定位过程中常用的手段，关于

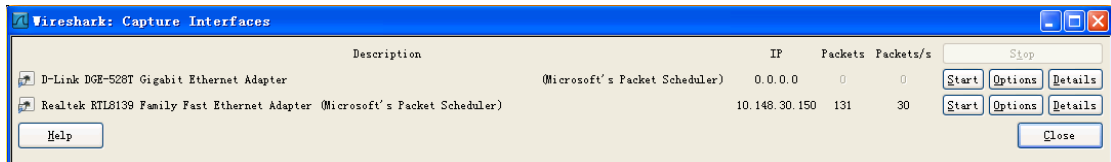
Wireshark的详细选单和用法请参阅Wireshark帮助手册。

启动Wireshark，在主界面的左上角即是抓包相关的选项。

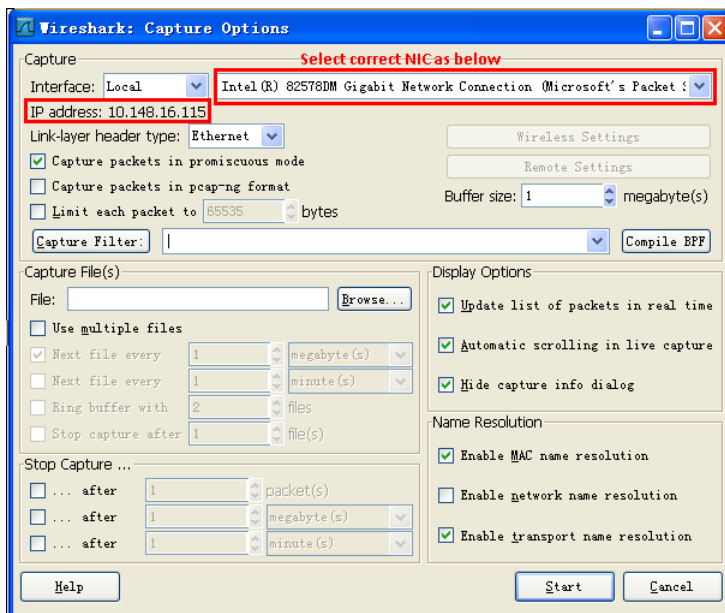


通常按照如下两种方法启动Wireshark抓包：

点击Interface List，或者选择菜单Capture -> Interfaces，或者按快捷键Ctrl+I，选择网卡启动抓包。当系统中存在多个网卡时，需要根据显示的IP地址选择正确的网卡然后点击Start以默认选项启动抓包。如果需要设置抓包选项，请点击对应的Options按钮。



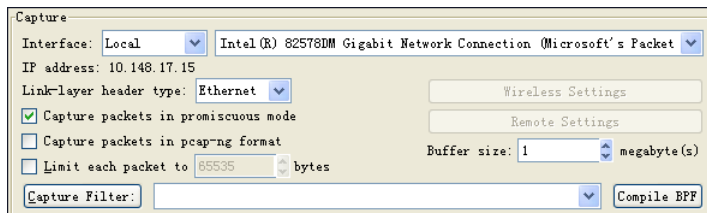
也可以在主界面点击Capture Options，或者选择菜单Capture -> Options，或者按快捷键Ctrl+K启动抓包选项设定，按需设定选项后再启动抓包。



停止抓包：按工具栏的停止键，或者选择菜单Capture -> Stop，或者按快捷键Ctrl+E停止抓包。

3.1.1 Wireshark 抓包选项

➤ Capture 区



Interface项：第一个下拉框可以选择是本地抓包还是远程抓包，一般设定Local就可以了，第二个下拉框会显示本地机器可供抓包的网卡，根据实际需要选择正确的网卡即可，选择网卡后，下一行的IP Address会显示该网卡绑定的IP地址（可能会有多个），可以利用IP地址来找到正确的网卡。

Link-layer header type：保持默认的Ethernet即可。

Wireless Settings和Remote Settings：一般不用。

Capture packets in promiscuous mode：在混杂模式下抓包。混杂模式：抓本网卡能接收到的所有数据包，不管是不是送给自己的。建议开启。

Capture packets in pcap-ng format：Wireshark的下一代抓包文件格式，当前版本尚在开发中，不建议开启。

Limit each packet to：限制抓包包长，截取对应字节数的包头部分。在抓包数据量很大时开启可以有效减小文件大小，Client/Server设置100Bytes，S1口设置150Bytes是比较安全的值。如果S1口开启了安全加密，不应该限制抓包包长。

NOTE

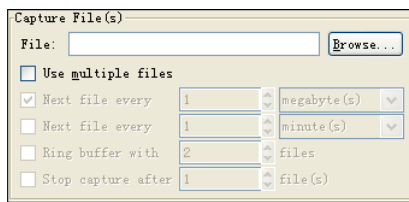
S1口开启安全加密的情况下，由于ESP协议封装除了报头以外，还存在报尾，这个时候不能进行抓包包长限制，否则Wireshark的解析功能不能正常工作。

Buffer Size：抓包文件被保存到硬盘之前暂存的内核缓存大小。如果发现Wireshark抓包时出现丢包，可以将这个值调大。

Capture Filter：捕捉滤波器。在抓包时进行过滤，只抓取满足设定规则的包，如果能明确抓包条件，建议使用。详细设定参见3.1.2。

Compile BPF：一般不用。

➤ 抓包文件存储



File: 保存抓包文件到用户指定的文件位置。如果不指定，Wireshark将保存抓包数据到一个临时文件中。

Use multiple files: 保存Wireshark抓包文件到多个文件中，防止单个抓包文件超大。当需要长时间抓包，或者抓包流量非常大的时候建议使用。下面是多个文件分割的条件：

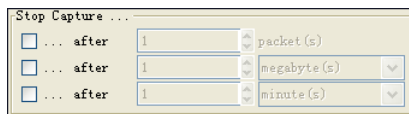
第一个选项，达到一定的大小即转存文件；

第二个选项，达到一定的时间即转存文件；

第三个选项，文件重复覆盖，达到设定的文件个数时开始从头覆盖旧文件；

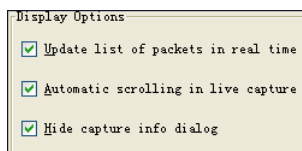
第四个选项，保存文件的最大数量，达到此文件数量即停止抓包。

➤ 自动停止抓包选项



顾名思义，达到设定的包数量/字节数/时间即自动停止抓包。

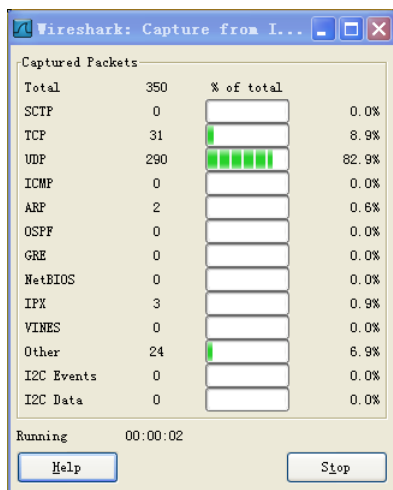
➤ 显示选项



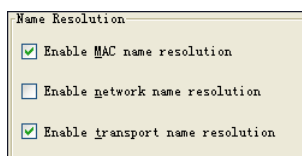
Update list of packets in real time: 实时更新抓包，如果不勾选，抓包数据在停止抓包后才会显示出来。

Automatic scrolling in live capture: 抓包时自动滚屏。

Hide capture info dialog: 是否在抓包时隐藏如下信息框：



名称解析



Enable MAC name resolution: 主要是否允许根据MAC地址解析制造商

Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
 Ethernet II, Src: HuaweiTe_0f:cf:f7 (28:6e:d4:0f:cf:f7), Dst: Universa_0f:79:96 (00:24:7e:0f:79:96)
 Internet Protocol Version 4, Src: 10.72.133.110 (10.72.133.110), Dst: 10.148.17.15 (10.148.17.15)

Enable network name resolution: 是否允许根据IP地址通过DNS解析主机名

No.	Time	Source	Destination	Protocol	Length
1	0.000000	10.148.17.15	hx-in-f99.1e100.net	ICMP	74
2	5.219751	10.148.17.15	hx-in-f99.1e100.net	ICMP	74

Enable transport name resolution: 是否允许根据TCP/UDP端口解析知名协议

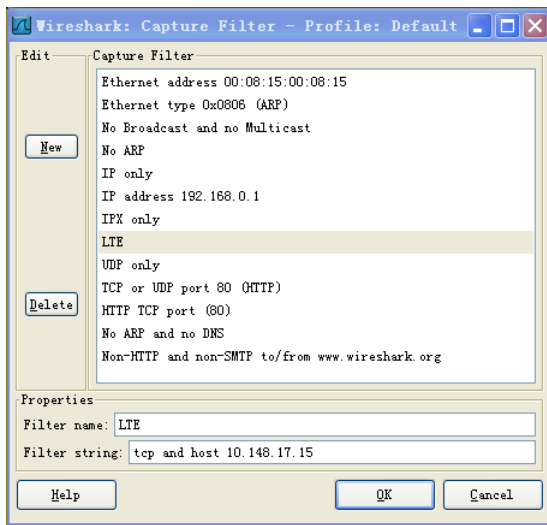
Transmission Control Protocol, Src Port: sunproxysadmin (8081), Dst Port: csd-monitor (3072), Seq: 30831, Ack: 81, Len: 26
 Data (26 bytes)

3.1.2 捕捉过滤器

Wireshark的捕捉过滤器使用libpcap滤波语言，主要是通过and/or/not三个连接符将各个原语表达式连接起来，其格式如下：

[not] primitive [and|or [not] primitive ...]

经常使用的过滤表达式也可以点击Capture Filter后保存下来，方便以后作为固定规则使用。



下面举例说明一些常用的过滤规则：

➤ 按地址过滤

host x.x.x.x 不区分源IP还是目的IP，只要包含此IP的数据包全部过滤出来；

src host x.x.x.x 按源IP过滤；

ether host yy:yy:yy:yy:yy:yy 按MAC地址过滤；

➤ 按协议过滤

直接输入协议类型即可，如tcp, udp, ip, icmp, arp等等；

➤ 按端口过滤

port 8080 过滤8080端口的数据；

3.2 使用其它软件采集数据

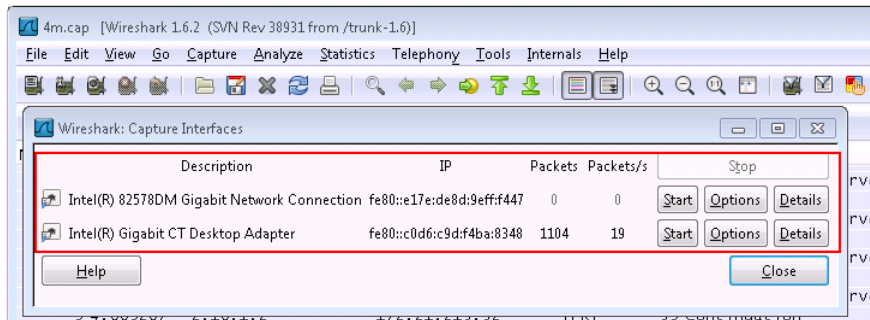
Wireshark支持丰富的文件格式，具体支持的文件格式可以在帮助文档中查阅。由于Wireshark通过pcap实现抓包，因此可以利用同样使用pcap的程序来采集数据再导入到Wireshark中进行分析，譬如tcpdump、CA NetMaster、Microsoft Network Monitor。

3.2.1 Windows 7 下 Wireshark 抓包找不到 Huawei E398 虚拟网卡，利用 MS Network Monitor 采集数据

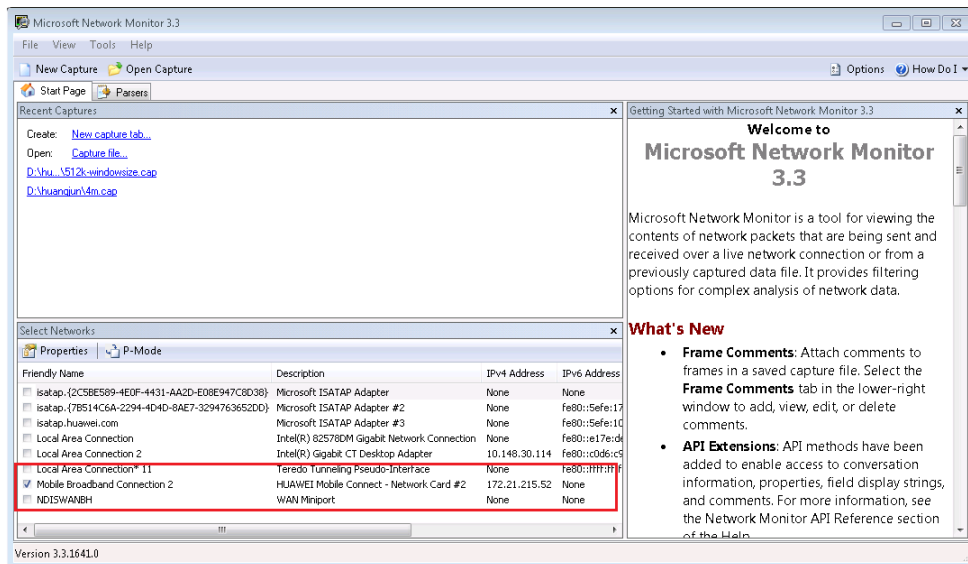
Win7下E398使用WWAN（Wireless Wide Area Network无线广域网）驱动会注册为移动网络设备，这时候Wireshark可能不能识别E398的网络接口导致无法使用Wireshark来抓包。由于Wireshark支持多种文件格式的解析，遇到这种情况可以使用MS Network Monitor

来抓包再导入Wireshark进行分析。

Wireshark无法识别E398对应的网络接口：



使用MS Network Monitor则没有问题：

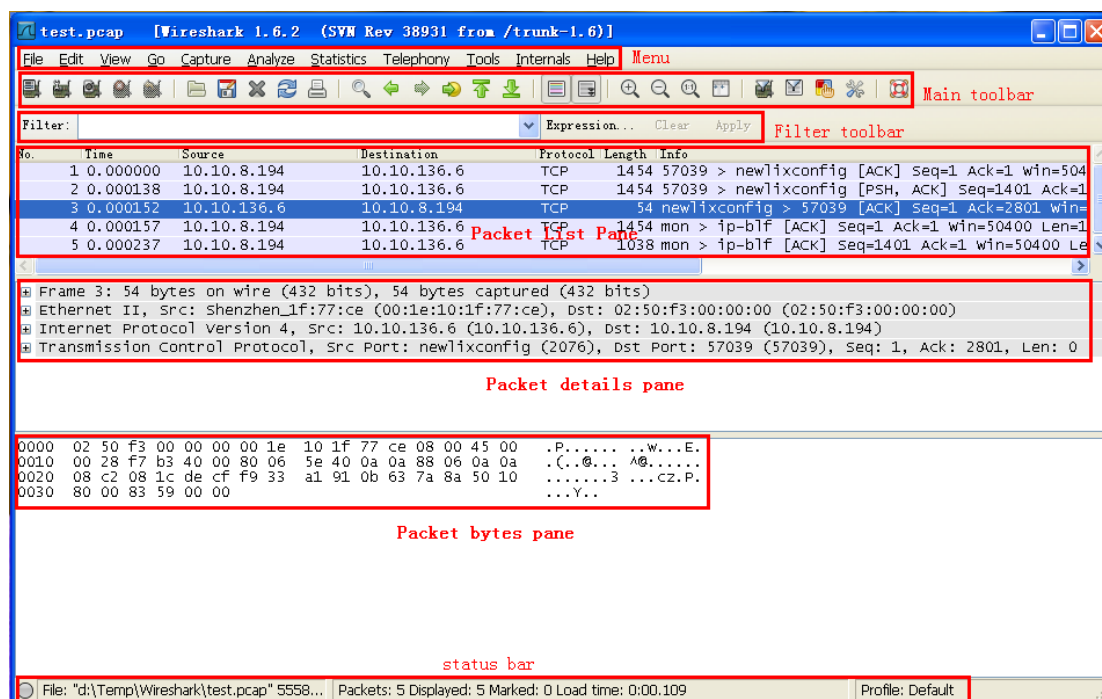


MS Network Monitor的使用比较简单，安装完成后启动程序，选择E398对应的网络接口，点击New Capture然后点击Start即可开始抓包。还可以通过菜单Tools -> Options做类似Wireshark的抓包选项基本配置。

4 软件设置

4.1 Wireshark 基本界面

如下图，Wireshark的基本界面如下，分为Menu（菜单栏），Main toolbar（主工具栏），Filter toolbar（显示过滤器工具栏），Packet list pane（包列表显示区），Packet details pane（包协议解析区），Packet bytes pane（二进制码流区），status bar（状态栏）七个部分。



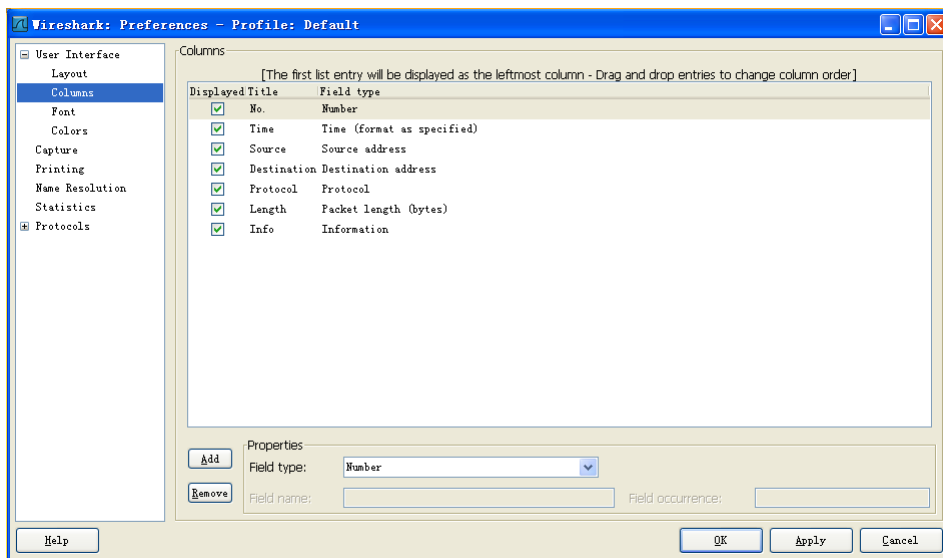
对于Packet list pane（包列表显示区），直接点击每个Column的title可以对该栏进行升序或者降序排列。

4.2 参数设置

Wireshark的设置选项位于菜单Edit -> Preferences，分User Interface、Capture、Printing、Name Resolution、Statistics、Protocols几大部分。各项参数的设置可以查阅Wireshark帮助手册，下面重点介绍帮助手册中没有但是我们常用的几个技巧：

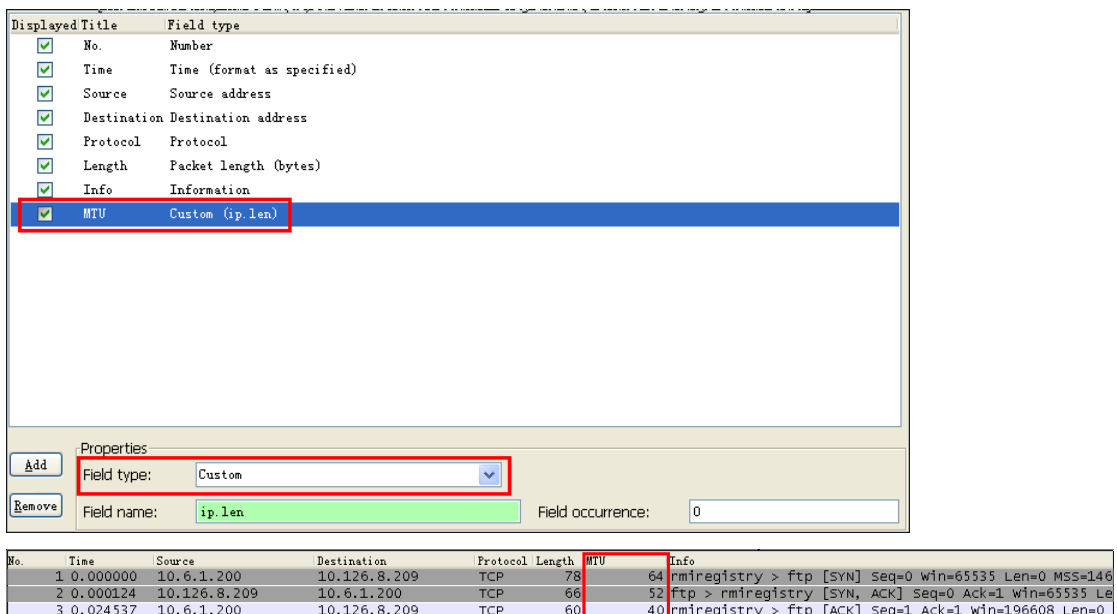
4.2.1 如何配置 Packet list pane（包列表显示区）显示项

Wireshark在包列表显示区默认显示序号、时间、源/目的IP、协议、总包长和信息等7项，这是在菜单Edit -> Preferences -> User Interface -> Columns里配置的。



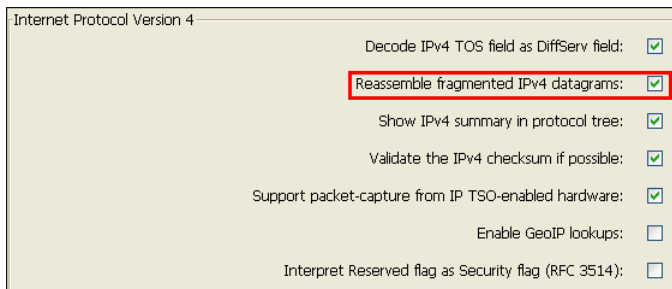
某些情况下，我们希望将自己常用的观察项也直接显示到包列表显示区，方便直接观察值。

点击Add按钮增加一个Column，直接单击New Column可以修改名称，比如我们要观察MTU，那么输入MTU。同时，在选中该行的情况下，点击属性栏的Field type下拉框，这里可以选择各种观察项，如果你的观察项不在选择列表中，那么选择Custom，这时下面的Field name会变成可编辑状态，输入你要观察的内容，比如我们要观察MTU，那么输入ip.len，本栏的语法规则同显示滤波器的语法是一样的，可以参考6.2节。Field occurrence栏保持0即可列出全部符合规则的事件。点击OK后，包列表显示区就增加了我们想要的字段。



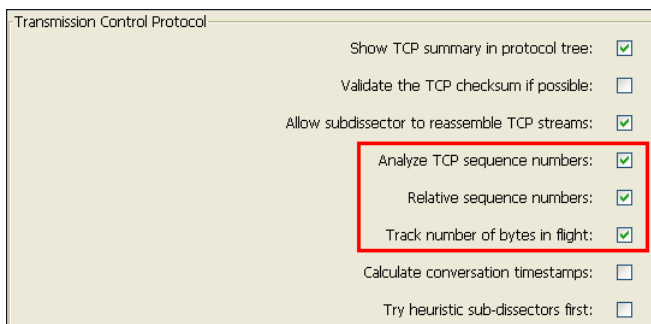
4.2.2 IP 协议配置

点击菜单Edit -> Preferences -> Protocols -> IPv4可以对IP协议的解析进行配置。各配置项基本可以通过查看项目名即可知道意义，这里需要重点关注Reassemble fragmented IPv4 datagrams这一项，如果抓包中存在分片包，请务必勾选此项，Wireshark可以直接对分片包进行重组后显示。



4.2.3 TCP 协议配置

菜单Edit -> Preferences -> Protocols -> TCP。重点注意三项，Analyze TCP sequence numbers，此项务必勾选，否则显示过滤器中所有tcp.analysis开头的滤波选项都将失效；relative sequence numbers，此项控制TCP分析是使用相对序列号还是绝对序列号，需要根据不同的场合进行切换；Track number of bytes in flight，勾选此项可以用来追踪链路中正在进行传输的包容量，以此可以用来近似跟踪发送缓冲的使用情况。



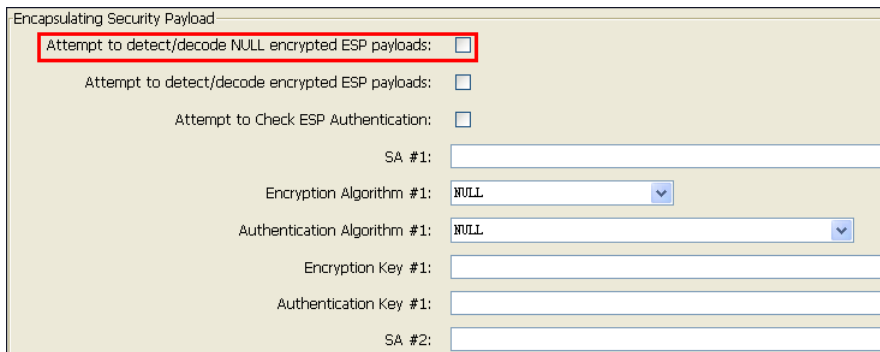
NOTE

Relative sequence，相对序列号。TCP 使用 4 个字节来标识其传送数据的字节流，由于每次建立 TCP 链接时，其 ISN (Initial Sequence Number) 可以近似理解为由操作系统随机生成，并非从 0 开始，此时这 4 个字节的序号我们可以理解为绝对序列号。由于使用绝对序列号并不方便，Wireshark 将一个新的 TCP 链接所有的序号都减去 ISN，得到一个从 0 开始的序号，称为相对序列号。

当分析一个从建链开始的单独 TCP 链接时，使用相对序列号较为方便；当需要进行两个并非从头开始抓包的抓包文件对比时，使用绝对序列号比较方便。

4.2.4 ESP 协议配置

在LTE中，当传输S1口进行了安全加密的情况下，抓包文件通常无法分析。Wireshark 提供了分析ESP加密文件的能力，在菜单Edit -> Preferences -> Protocols -> ESP中进行配置。在S1口设置为ESP空加密的情况下，需要将Attempt to detect/decode NULL encrypted ESP payloads选项勾选，才能对ESP协议进行处理。



5 数据处理

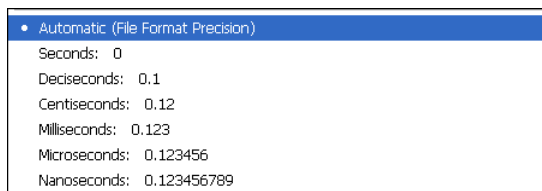
5.1 常规技能

5.1.1 时间设置

在抓包的时候，对每一个包，Wireshark会利用WinPcap记录一个时间戳，这个时间来源于操作系统内核，精确到微秒，并且写在抓包文件内，所以我们可以利用Wireshark作精确度相当高的时间分析。

➤ 时间精度设置

菜单View -> Time Display Format，如图所示，可以选择各种时间精度。除非系统增添了特别的专用计时设备，Wireshark统计到微秒的精度。



➤ 时间计算方式设置

菜单View -> Time Display Format，如图所示，可以设置时间显示格式是从年开始

还是从天开始。也可以设置每个包的时间戳是否都是相对于UTC标准时间，还是相对于第一个抓包的时间差，或者是相对于上一个抓包的时间差，再或者是相对于上一个显示包的时间差。

Date and Time of Day: 1970-01-01 01:02:03.123456	Ctrl+Alt+1
Time of Day: 01:02:03.123456	Ctrl+Alt+2
Seconds Since Epoch (1970-01-01): 1234567890.123456	Ctrl+Alt+3
• Seconds Since Beginning of Capture: 123.123456	Ctrl+Alt+4
Seconds Since Previous Captured Packet: 1.123456	Ctrl+Alt+5
Seconds Since Previous Displayed Packet: 1.123456	Ctrl+Alt+6

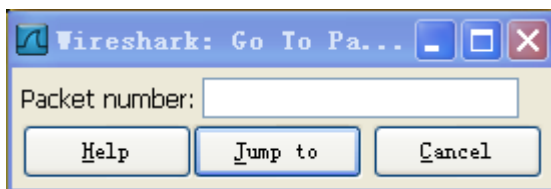
➤ 包时间参考点

选中某个包，右键选择**Set Time Reference**，可以将某个包设为时间参考点，这样此后的包都将以这个包的时间作为时间原点来计算时间差，这项功能在将某个**request**设为参考点统计响应时间时特别有用。

可以在一个抓包文件中设置多个时间参考点。再次选择此项则为取消。

5.1.2 如何快速找到某个包

在知道包序号的情况下，点击菜单**GO -> Go to Packet**或者直接按快捷键**Ctrl + G**，然后输入包序号，可以直接跳转到对应的包处。



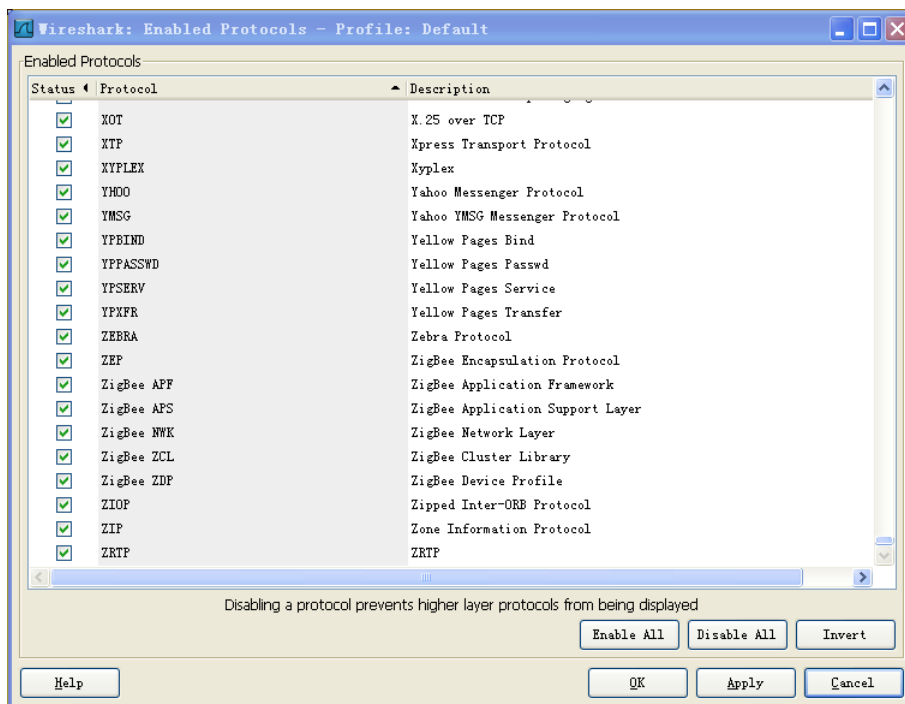
5.1.3 指定协议解析

Wireshark对于每个协议都有一个自己的解析器，Wireshark在打开文件时会试着去查找最合适的解析器来解包，但是这个过程可能出错，比如用一个不常用的端口代替80端口进行HTTP业务。这种情况下，我们需要对抓包文件进行指定协议解析。

有两种方法来控制解析协议：一是禁止某种协议解析器；二是指定抓包文件被解析为某种协议。

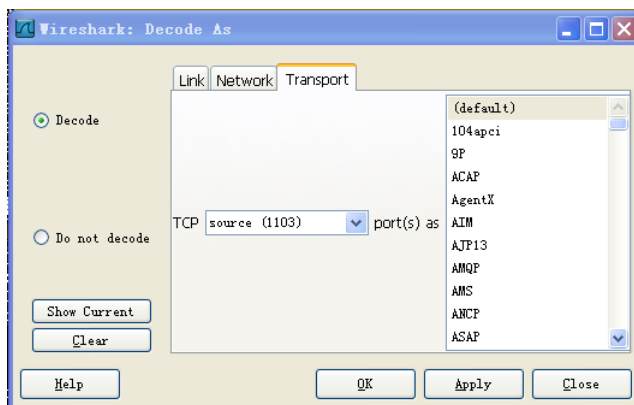
➤ 禁止某种协议解析器

菜单**Analyze -> Enabled Protocols**，这里列出了所有Wireshark支持的协议列表，如果不想抓包文件被解析为某种协议，点击取消对应协议前方的勾即可。



➤ Decode As

菜单Analyze -> Decode As或者在包列表上直接点击右键选择Decode As，这里可以直接指定Link、Network、Transport三层使用的具体协议。



5.1.4 如何自动重组分片包

按照[4.2.2](#)配置即可。

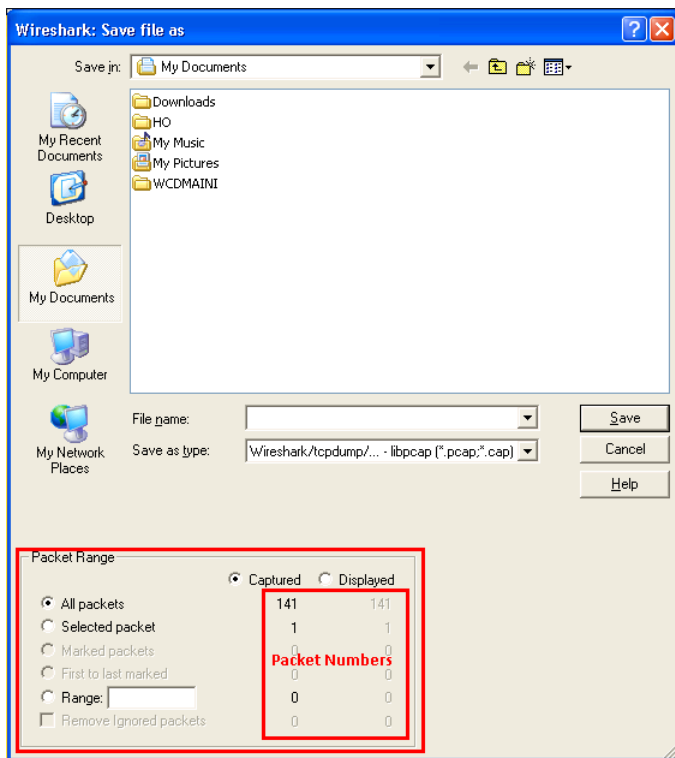
5.2 文件处理

5.2.1 仅保存需要的数据

有时候Wireshark抓包得到的数据并非全部都是我们想要的，还可能存在很多杂包，或

者我们只想截取其中的一部分进行分析,同时,LTE数据量巨大,抓包得到的数据通常很大,需要作进一步处理,化大文件为小文件,提高每次打开文件分析问题的效率。下面介绍如何通过Wireshark的Save As功能保存需要的数据。

选择菜单File -> Save As打开对话框, Packet Range这部分就是我们要关注的内容。



当设置不同的条件时,中心块的数字会显示符合条件的包数量。横排是条件,纵向另有两个约束条件, **Captured**表示保存整个抓包文件范围内满足条件的包, **Displayed**表示仅保存显示在Wireshark界面上满足条件的包,造成这种差异的原因是因为我们可以使用显示滤波器对抓包进行过滤,只在界面上显示出一部分抓包结果,关于显示滤波器的内容请参见5.1节。

- **All packets:** **Captured**表示所有抓包, **Displayed**表示滤波后显示的抓包;
- **Selected Packet:** 由于Wireshark同一时间只能选择一个包,所以这项的意义就是保存选择的那个包, **Captured**和**Displayed**都一样。
- **Marked Packets:** **Captured**表示保存所有标记的包, **Displayed**表示保存过滤后标记的包。如果未标记任何包此项将成灰色不可选状态。

NOTE

Marked Packet: 在 Wireshark 主界面选中某个包,右键选择 **Mark Packet**,可以标记某个包。被标记的包会以黑色背景高亮显示,再次选择 **Mark Packet** 可以取消标记。可以标记多个包。

- **First to last Marked:** 在标记了多个包的情况下，保存从第一个标记的包到最后一个标记的包之间的所有包，可以用这种方法来保存一段包。
- **Range:** 保存包序号在某段范围内的包，可以标识多段范围，范围之间彼此用“,”隔开，比如“1,6-8”表示保存包序号1, 6, 7, 8的四个包，还可以用“20-”表示从包序号20开始直到抓包末尾的包。
- **Remove Ignored Packets:** 效果同反向选择，被忽略的包被删除，保存其它的包。

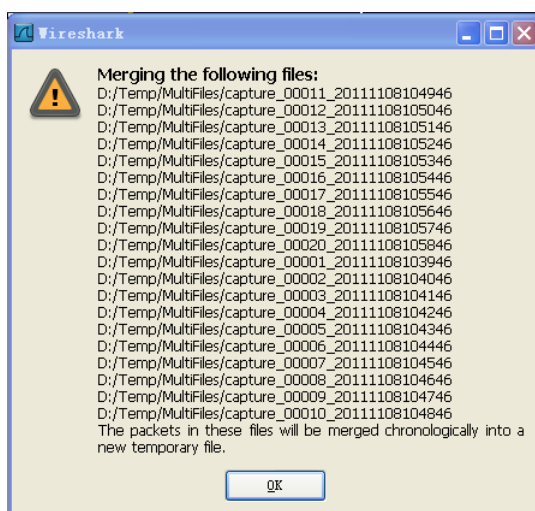
NOTE

Ignored Packet: 在 Wireshark 主界面选中某个包，右键选择 **Ignore Packet**，可以略过某个包。被略过的包会显示为空白（包序号仍占一行），再次选择 **Ignore Packet** 可以取消。可以同时略过多个包。被忽略的包同删除的效果一样，因此略过某些包可能触发 Wireshark 丢包判断。

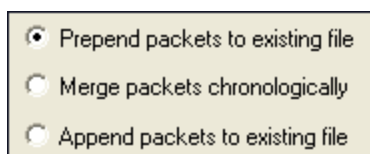
5.2.2 多文件合并

抓包过程中使用多文件可能产生多个连续的文件，在使用Wireshark进行分析时可能需要将这些文件进行合并，常用的多文件合并方法有两种：

- **文件拖拽:** 选中要合并的文件，直接全部一起拖进Wireshark即可。Wireshark会提示要合并文件，直接点击OK即完成。合并按照包记录的时间先后顺序排序，可以将合并后的文件作为一个文件单独存储。



- **Merge功能:** 必须首先打开一个抓包文件，然后才能激活菜单File -> Merge功能。

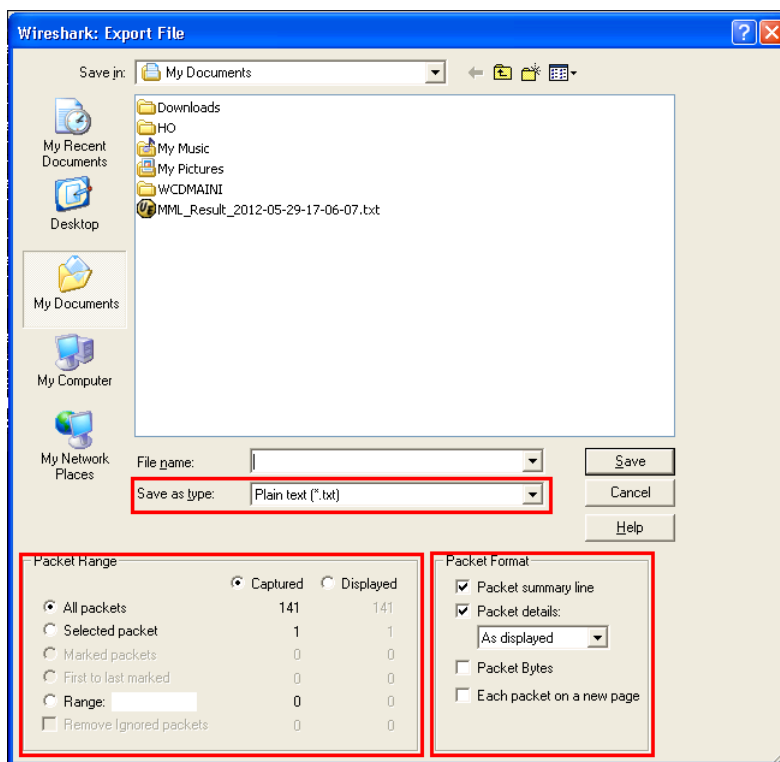


- 第一项， 将要合并的抓包文件放在当前文件之前；
- 第二项， 按照抓包的时间顺序合并；
- 第三项， 将要合并的抓包文件追加到当前文件之后；

5.2.3 转换文件格式输出

有时候我们需要将Wireshark解析的数据结果输出成别的文件格式以供阅读、打印或者其它程序处理，这个时候需要用到Wireshark的文件格式转换功能。

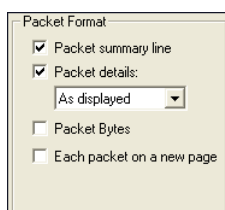
打开一个抓包文件，点击菜单File -> Export -> File启动Wireshark的文件输出功能。



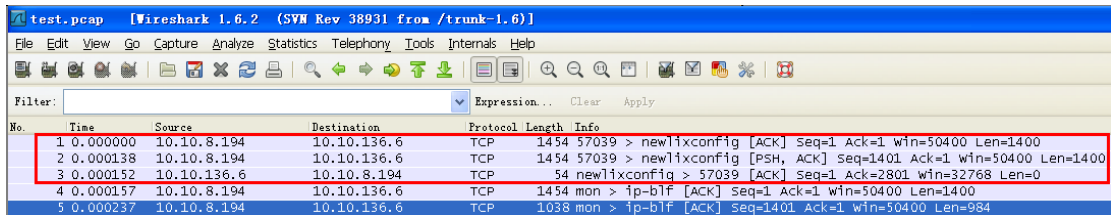
保存类型项，Wireshark默认输出文本格式，但同时还提供其它5种格式的文件类型可供选择，具体文件格式直接点击下拉框可见。

Packet Range，同4.1.1介绍的内容完全一样，这里不再作介绍。

Packet Format，这里重点介绍一下。



- **Packet summary line:** 是否允许输出包概要信息。包概要信息就是我们在主界面 **Packet list pane**（包列表显示区）中看到的信息，如图：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.8.194	10.10.136.6	TCP	1454	57039 > newlixconfig [ACK] Seq=1 Ack=1 Win=50400 Len=1400
2	0.000138	10.10.8.194	10.10.136.6	TCP	1454	57039 > newlixconfig [PSH, ACK] Seq=1401 Ack=1 Win=50400 Len=1400
3	0.000152	10.10.136.6	10.10.8.194	TCP	54	newlixconfig > 57039 [ACK] Seq=1 Ack=2801 Win=32768 Len=0
4	0.000157	10.10.8.194	10.10.136.6	TCP	1454	mon > ip-b1f [ACK] Seq=1 Ack=1 Win=50400 Len=1400
5	0.000237	10.10.8.194	10.10.136.6	TCP	1038	mon > ip-b1f [ACK] Seq=1401 Ack=1 Win=50400 Len=984

- **Packet details:** 下拉框有三个选项，**All collapsed**将**Packet details pane**（包协议解析区）的内容全部收拢显示，**As displayed**将包协议解析区的内容按当前展开的状态显示，**All expanded**将包协议解析区的内容全部展开显示。
- **Packet Bytes:** 是否允许显示**Packet bytes pane**（二进制码流区）的内容。
- **Each packet on a new page:** 每个包内容是否分页。

5.3 Wireshark 自带的命令行工具

Wireshark除GUI界面的主程序外，还包括一些支持命令行调用的小工具，这些小工具可能会对分析、处理某类问题时特别有用，它们包括：**tshark**，Wireshark的命令行简化版；**tcpdump**，一个命令行抓包工具；**dumpcap**，另一个命令行抓包工具；**capinfos**，查看Wireshark抓包文件信息的小工具；**rawshark**，支持实时抓包分析的简化版工具；**editcap**，pcap文件编辑工具；**mergcap**，合并多个抓包文件的工具；**text2pcap**，转换16进制的抓包文件成pcap格式。这些工具的用法都可以在Wireshark帮助手册中查询到，这里就不一一介绍，下面仅介绍我们可能会用到的几个小工具。

5.3.1 如何使用 Wireshark 命令行工具

依次在Windows中点击开始->运行，输入cmd回车启动Windows命令行，然后进入Wireshark安装目录的路径，此时即可调用Wireshark自带的各种命令行工具，输入这些工具对应的命令格式即可执行。如图：

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\User>cd C:\Program Files\Wireshark
C:\Program Files\Wireshark>tshark
Capturing on Marvell Yukon Ethernet Controller (Microsoft's Packet Scheduler)
0.000000 10.141.10.51 -> 10.148.16.115 TCP 54 ms-wbt-server > comotionback [ACK]
K1 Seq=1 Ack=1 Win=32703 Len=0
0.067057 10.148.16.115 -> 10.141.10.51 TCP 60 comotionback > ms-wbt-server [ACK]
K1 Seq=1 Ack=1 Win=63399 Len=0
0.108257 10.148.16.115 -> 10.141.10.51 T.125 102 T.125 payload
0.171952 10.141.10.51 -> 10.148.16.115 TPkt 96 Continuation
0.366853 10.148.16.115 -> 10.141.10.51 TCP 60 comotionback > ms-wbt-server [ACK]
K1 Seq=49 Ack=43 Win=63357 Len=0
1.133052 10.141.10.41 -> 10.141.10.255 NBNS 92 Name query NB WPAD<00>
1.575653 Hangzhou_14:ba:e0 -> Spanning-tree<for-bridges>_00 STP 120 MST. Root
= 32768/0/00:23:89:14:ba:d0 Cost = 0 Port = 0x8001
1.621847 Huawei1e_38:1b:48 -> Broadcast ARP 60 Who has 10.141.10.221? Tell
10.141.10.1
1.883247 10.141.10.41 -> 10.141.10.255 NBNS 92 Name query NB WPAD<00>
2.497009 10.141.10.51 -> 10.148.16.115 TPkt 70 Continuation
2.632646 10.141.10.41 -> 10.141.10.255 NBNS 92 Name query NB WPAD<00>
2.680446 10.148.16.115 -> 10.141.10.51 TCP 60 comotionback > ms-wbt-server [ACK]
K1 Seq=49 Ack=59 Win=63341 Len=0
```

5.3.2 利用 editcap 分割文件

Editcap 可以实现对 PCAP 文件的各类操作，如输出选择的数据包，删除数据包，截取数据包，分割抓包文件等，部分功能可以在 GUI 界面中实现，这里介绍 editcap 命令行特有的功能。

➤ 文件分割

Editcap -c xxxx srcfile dstfile

xxxx 表示分割后每个文件包含多少个数据包，srcfile 表示源文件名，dstfile 表示目标文件名。文件名均可以包含路径。

Editcap -i yyyy srcfile dstfile

yyyy 表示分割后每个文件包含几秒的数据，srcfile 表示源文件名，dstfile 表示目标文件名。文件名均可以包含路径。

如下面例子，一个包含 52s 数据的 pcap 包文件被 editcap 分割成了 10s 一个文件。

```
C:\WINDOWS\system32\cmd.exe

C:\Program Files\Wireshark>editcap -i 10 d:\temp\server.pcap d:\temp\serverp
rocess.pcap.

C:\Program Files\Wireshark>
```

SERVER	pcap		
serverprocess	pcap_00000_20110420053646	9,299,585	11-04-19 23:53 -a
serverprocess	pcap_00001_20110420053657	1,166	11-11-09 17:06 -a
serverprocess	pcap_00002_20110420053706	2,468,869	11-11-09 17:06 -a
serverprocess	pcap_00003_20110420053716	2,212,490	11-11-09 17:06 -a
serverprocess	pcap_00004_20110420053726	2,343,346	11-11-09 17:06 -a
serverprocess	pcap_00005_20110420053736	1,961,304	11-11-09 17:06 -a
serverprocess	pcap_00005_20110420053736	312,530	11-11-09 17:06 -a

6 数据分析

6.1 Wireshark 数据分析基本知识

以下内容着重介绍**Wireshark**在**LTE**数传分析中的应用，我们假定读者已经具备**TCP/IP**协议以及**LTE**基本知识。

6.1.1 Wireshark 基本标识

Wireshark根据安装的协议解析器自动为每个包选择最合适的协议进行解析，通常分为5个部分，如图：

```
Frame 2415: 1514 bytes on wire (12112 bits), 100 bytes captured (800 bits) on interface 0
Ethernet II, Src: Cisco_O6:c4:40 (00:15:fa:06:c4:40), Dst: HuaweiTe_c6:7f:3a (00:18:82:c6:7f:3a)
Internet Protocol Version 4, Src: 10.6.1.200 (10.6.1.200), Dst: 10.126.8.209 (10.126.8.209)
Transmission Control Protocol, Src Port: adobeserver-2 (1103), Dst Port: dyn-site (3932), Seq: 2177289, Ack: 1, Len: 1460
FTP Data
```

第一层是Wireshark抓包时记录每个包的时间戳，这是不包含在抓包内容当中的，在二进制码流解析区也找不到对应的二进制码流。

其余四层分别对应OSI七层模型当中的数据链路层、IP层、传输层和应用层。如果没有数据，比如ping，允许部分层内容空缺。

Wireshark会根据协议解析每一个字段，这里面会有一些内容说明，如下图。

```
Transmission Control Protocol, Src Port: adobeserver-2 (1103), Dst Port: dyn-site (3932), Seq: 2177289, Ack: 1, Len: 1460
Source port: adobeserver-2 (1103)
Destination port: dyn-site (3932)
[Stream index: 4]
Sequence number: 2177289 (relative sequence number)
[Next sequence number: 2178749 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x10 (ACK)
window size value: 32768
[Calculated window size: 4194304]
[window size scaling factor: 128]
```

以 () 号括起来的文字是Wireshark解析的备注，或者是二进制码流直接对应的数字，或者是Wireshark对这一字段的说明。

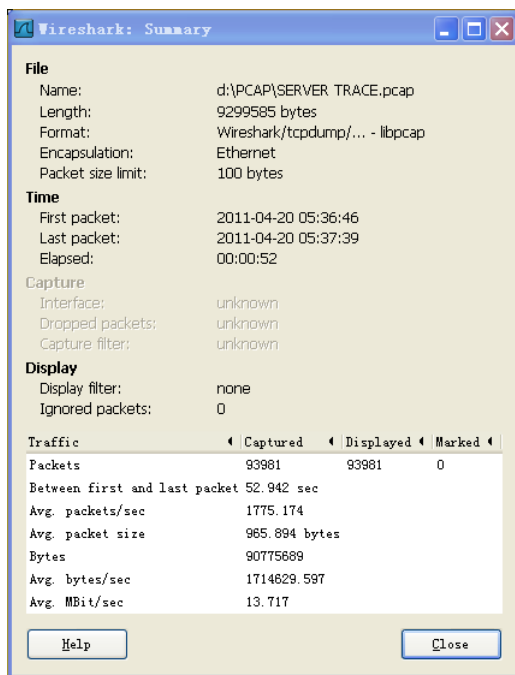
以 [] 号括起来的文字是Wireshark根据抓包内容作出的分析或者计算结果，和 () 号括起来的内容一样，它们都不是抓包文件二进制码流直接对应的解析结果。

没有被包含在任何括号的内容才是Wireshark直接解析协议字段得到的内容，它们是有二进制码流与之对应的。

6.1.2 抓包统计 Summary

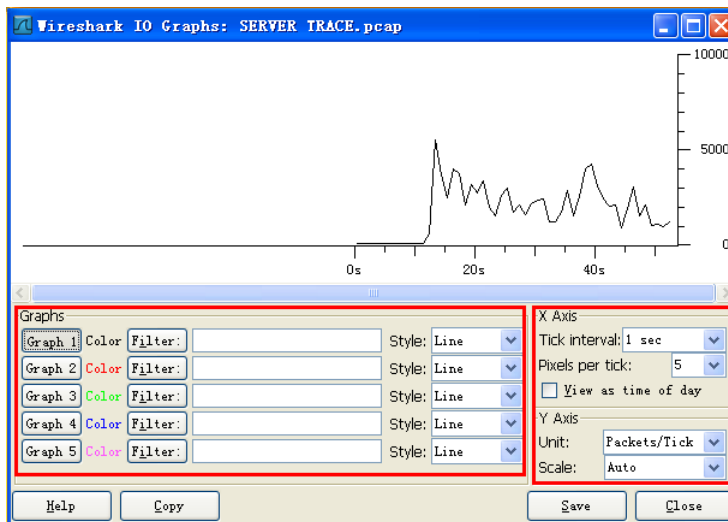
拿到一个抓包，我们如何查看这个抓包的一些基本信息，比如抓了多少个包，持续时间多长，总的字节数是多少，平均速率怎么样，可以使用菜单**Statistics -> Summary**做一个抓包基本信息统计。

此项统计在进行UDP灌包两点抓包统计丢包率，检验Dumeter等流量工具的正确性等测试时尤其有用。



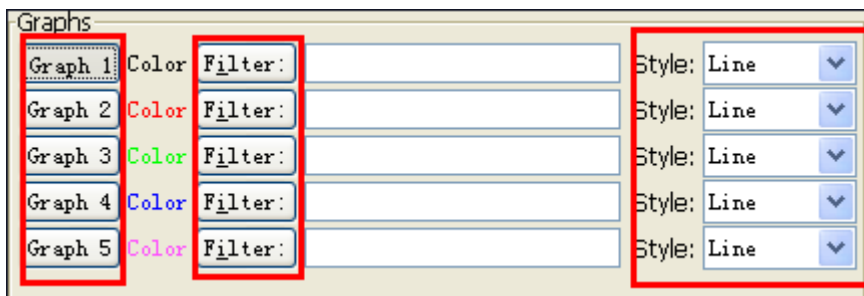
6.1.3 流量统计 IO Graphs

IO Graphs以图形形式显示当前抓包的数据量，这个数据量包括Wireshark抓下来的所有包在内。IO Graphs可以以不同间隔、单位和色彩来显示，点击**Statistics -> IO Graphs**启动IO流程图。



如上图，IO Graphs上半部分是图形显示区，下半部分左边是图形设置区，右边是坐标轴设置区，下面分别介绍。

Wireshark可以5种不同的色彩来分别显示不同的数据流。

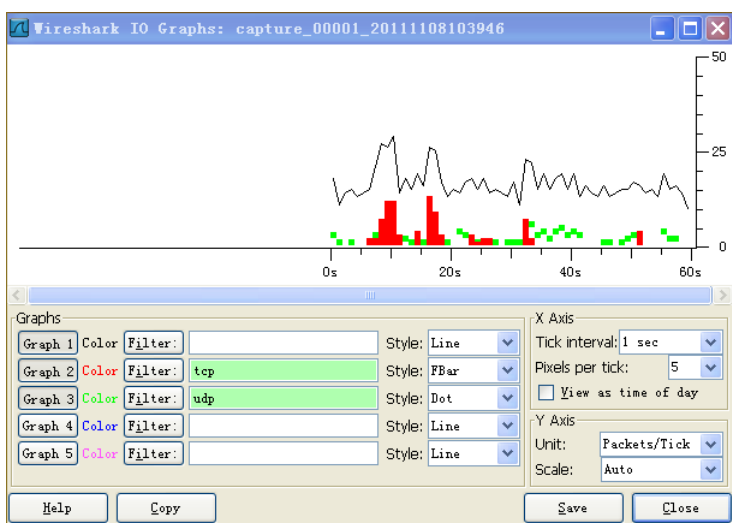


Filter: 可以对不同颜色显示的数据流进行设定，Filter的写法同显示滤波器（参见6.3）。

Style: 对现实的方式进行设置。

Graph: 按钮，按下进行该色的绘图，放开不对该色进行绘图。

如下图，对不同协议的包流量进行了区分：



X Axis:

Tick interval, 设定X轴的采样间隔, 最小可以设定**1ms**, 这在我们进行精细分析时非常有用。

Pixels per tick, 设定每一个采样间隔用多少像素来绘图, 可以设定**1, 2, 5, 10**, 越大图形越准确, 但是对系统要求越高。

View as time of day: 如勾选, 则X轴的单位将以**24小时制**的实际时间来显示。

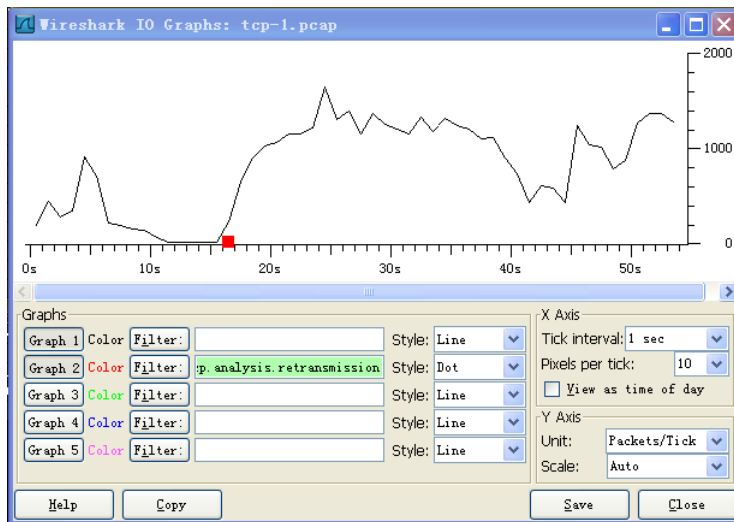
Y Axis:

Unit: 设定Y轴的单位, 可以是包数, 也可以是字节, 或者比特流。这里要注意的是, 单位的分母是**tick**, 要么需要根据X轴设定的一个**tick**表示什么才能换算成我们实际运用的单位。

Scale: Y轴的最大量程, 此处可以控制Y方向的放大缩小, 也可以以对数形式表示。

下面还有几个按钮, **Copy**可以将当前显示图形的数据**copy**出来存成**CSV**的格式供进一步分析; **Save**按钮可以保存当前显示的图形。

IO Graphs的应用堪称经典, 因为我们能把所有的显示过滤器放到**filter**一栏来作图, 所以利用**IO Graphs**几乎可以直观的观察到的任何东西。如下图, 我们利用图形来观察整个抓包中有多少重传, 当然也可以利用其它显示过滤器来观察其它任何能过滤的包。



NOTE

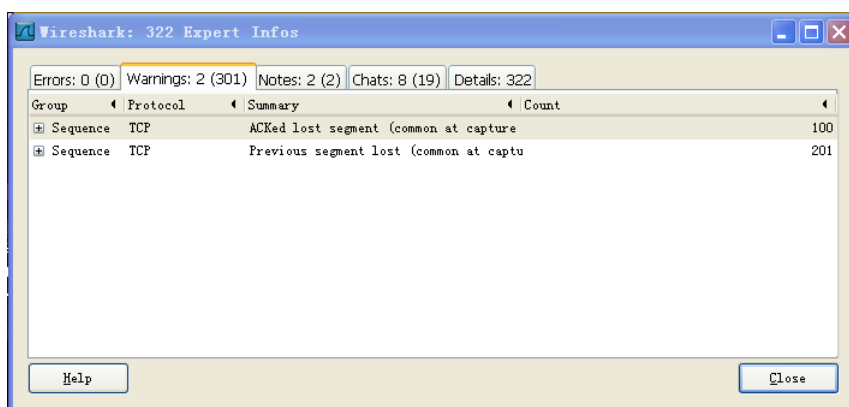
使用 IO Graghs 作图时注意选择 **Style**。如果抓包的数据量很大但是滤出的包很少时使用 **Line** 方式作图可能会很不明显，看不出异常，这个时候应试试别的 **Style**。

在 IO Graghs 图形上单击对应的包，Wireshark 主界面的包列表区会自动跳转到对应的位置。

6.1.4 汇总所有日志的专家信息 Expert Info Composite

Expert Info Composite可以说是Wireshark的告警管理模块，Expert Info Composite能显示Wireshark中所有的错误提示信息，以及建链包等含有关键信息的包。

在拿到一个抓包文件之后，可以使用菜单Analyze -> Expert Info Composite先大概评估一遍抓包文件中包含有哪些类型的错误提示信息需要核查，再针对具体的问题进行具体分析。



如上图，Wireshark将“uncommon”的包分成了4类，每一类代表了一种紧急度。

- **Errors:** 严重问题，错误的包或者Wireshark无法解析的包；
- **Warnings:** 警告问题，非正常通信的包，如分段丢失这类连接问题；
- **Notes:** 提示信息，可能是正常通信的一部分，如TCP重传、重复ACK等等都属于

这类;

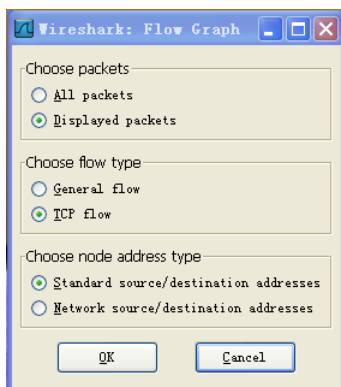
- Chats: 含有链路基本信息的包, 如TCP的SYN包;
- Details: 上述4类包的汇总。

当在Expert Info Composite中选中某个包之后, Wireshark主界面会直接跳转到对应的位置。

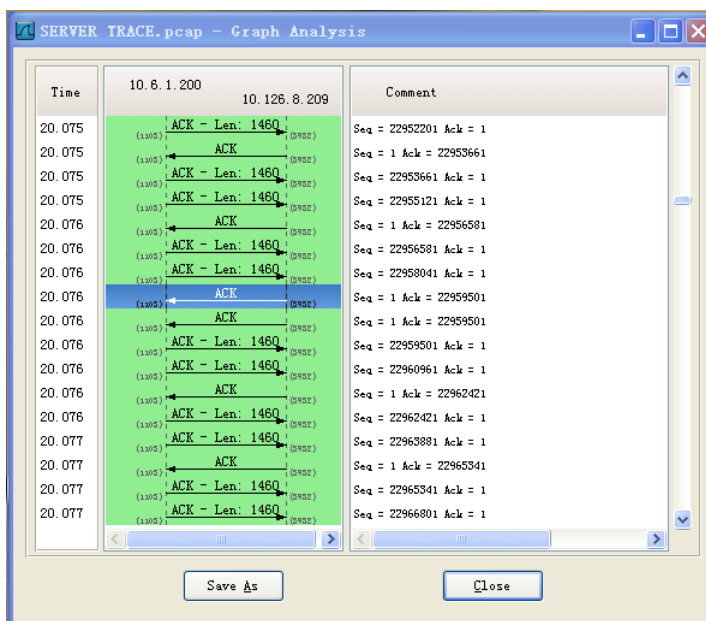
6.1.5 交互流图 Flow Graph

Flow Graph图用来进行流程分析很有用, 点击Statistics -> Flow Graph启动交互流图。

在生成具体的图形前Wireshark会要求做一些设定, Flow Graph的设定都比较简单, 一看便可明白。



点击OK后Wireshark会生成具体的流图, 这一步根据抓包文件的大小需要一定的处理时间。如下图:



左侧是时间，中间是根据协议对交互内容的分析结果，右侧是具体的细节。双击中间某个包，Wireshark可以立即在包显示区显示对应的包，非常方便。

6.2 TCP 基本参数分析

6.2.1 TCP 接收窗口

由于TCP真实的接收窗口计算需要用到窗口扩展因子WS，而WS仅仅带在TCP建链的SYN包中，所以要从Wireshark抓包中看出真实的接收窗口大小必须在进行TCP数传之前就开始抓包。

No.	Time	Source	Destination	Protocol	Length	MTU	Info
58	8.863318	10.26.192.18	89.9.162.1	TPKT	429		41> Continuation
59	8.872494	10.26.192.18	89.9.162.1	TPKT	70		56 Continuation
60	8.878848	89.9.162.1	10.26.192.18	TCP	60		40 hp-hcip-gwy > ms-wbt-server [ACK] Seq=213 Ack=4459 win=3
61	8.883839	89.9.162.1	10.26.192.18	TCP	60		40 hp-hcip-gwy > ms-wbt-server [ACK] Seq=213 Ack=4850 win=3
62	8.883850	89.9.163.6	10.26.192.18	TCP	60		40 5001 > syam-agent [ACK] Seq=1 Ack=1235 win=524288 Len=0
63	8.883877	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [ACK] Seq=1235 Ack=1 win=522720 Len=12
64	8.883893	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [ACK] Seq=2445 Ack=1 win=522720 Len=12
65	8.883905	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [ACK] Seq=3655 Ack=1 win=522720 Len=12
66	8.894819	89.9.163.6	10.26.192.18	TCP	60		40 5001 > syam-agent [ACK] Seq=1 Ack=3655 win=524288 Len=0
67	8.894834	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [ACK] Seq=4865 Ack=1 win=522720 Len=12
68	8.894846	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [ACK] Seq=6075 Ack=1 win=522720 Len=12
69	8.894857	10.26.192.18	89.9.163.6	TCP	1264		1250 syam-agent > 5001 [PSH, ACK] Seq=7285 Ack=1 win=522720 L
70	8.905822	89.9.163.6	10.26.192.18	TCP	60		40 5001 > syam-agent [ACK] Seq=1 Ack=6075 win=524288 Len=0
71	8.905826	89.9.163.6	10.26.192.18	TCP	60		40 5001 > syam-agent [ACK] Seq=1 Ack=8495 win=524288 Len=0

如图所示，在包列表区info栏里win=xxxxx这里带的值就是接收窗口的大小了。但是要注意区分源IP，比如在进行下载时，接收窗口是由接收端反馈的，那么要看源IP是接收窗口的行。源IP是发送端的行带的是发送端的接收窗口，不是发送窗口。

NOTE

1.6.2 版本以前的 Wireshark 有开关控制是否在计算接收窗口时使用窗口扩展因子，如果这个控制项没有打开，也会造成接收窗口看上去像没有优化过的值。

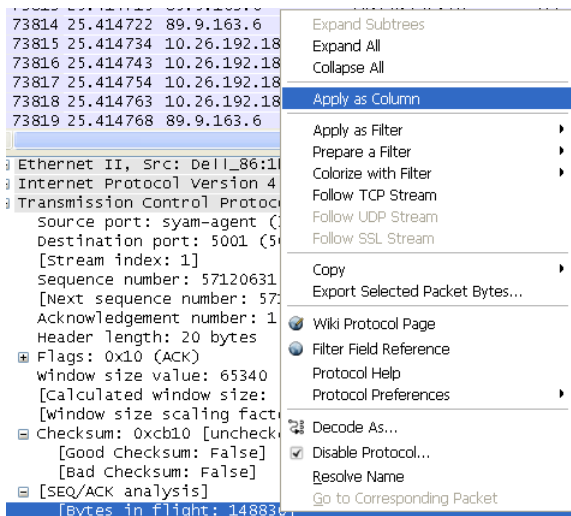
在菜单 Edit -> Preferences -> Protocols -> TCP 设置里找到 Relative sequence numbers and window scaling 项打上勾即可，此项和相对序列号是绑定在一起的，使用扩展因子时必须使用相对序列号。

6.2.2 TCP 发送窗口

发送窗口必须使用发送端的抓包文件才可以看到。

从BDP公式我们可以知道，理想情况下，发送窗口的大小可以近似用端到端的管道容量来逼近，新版的Wireshark提供了一个滤波器Bytes in flight来表示在网络中传输的字节数，可以利用这个字段来取得TCP链路的发送窗口。

选取一个数据包（不能是ACK）的Bytes in flight字段，右键执行Apply as Column，将Bytes in flight显示到包列表区，再点击Bytes in flight进行排序，找到最大的一个值，TCP发送窗口不小于这个值。

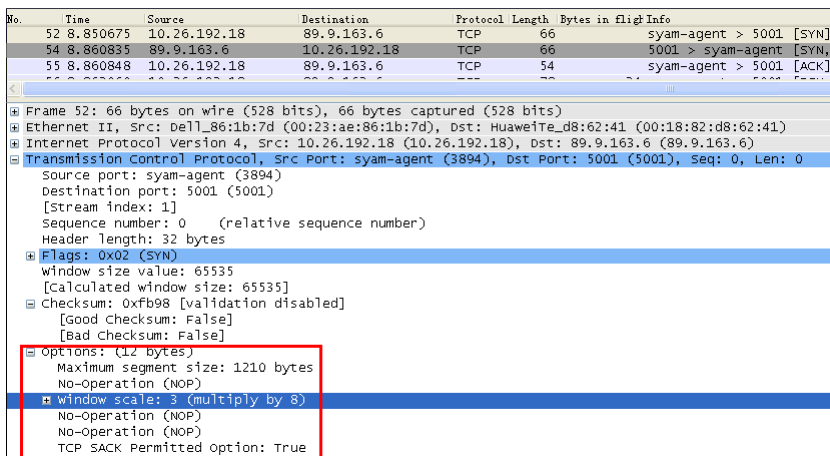


No.	Time	Source	Destination	Protocol	Length	MTU	Bytes in flight	Info
4128	9.384653	10.26.192.18	89.9.163.6	TCP	986	1270	524288	TCP window Full] syam-agent > 5001 [ACK]
4131	9.384673	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [PSH]
4134	9.384719	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4137	9.384792	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4140	9.384830	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4143	9.384885	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4146	9.384916	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4149	9.384939	10.26.192.18	89.9.163.6	TCP	986	972	524288	TCP window Full] syam-agent > 5001 [PSH]
4176	9.385481	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4179	9.385496	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4185	9.385619	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4188	9.385647	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4194	9.385715	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4197	9.385753	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4200	9.385766	10.26.192.18	89.9.163.6	TCP	1264	1250	524288	TCP window Full] syam-agent > 5001 [ACK]
4203	9.385795	10.26.192.18	89.9.163.6	TCP	986	972	524288	TCP Fast Retransmission] syam-agent > 5001 [ACK]

此方法需要保证抓包文件至少包含一段稳定期，否则得不到真实的TCP发送窗口。

6.2.3 其它 TCP 基本参数

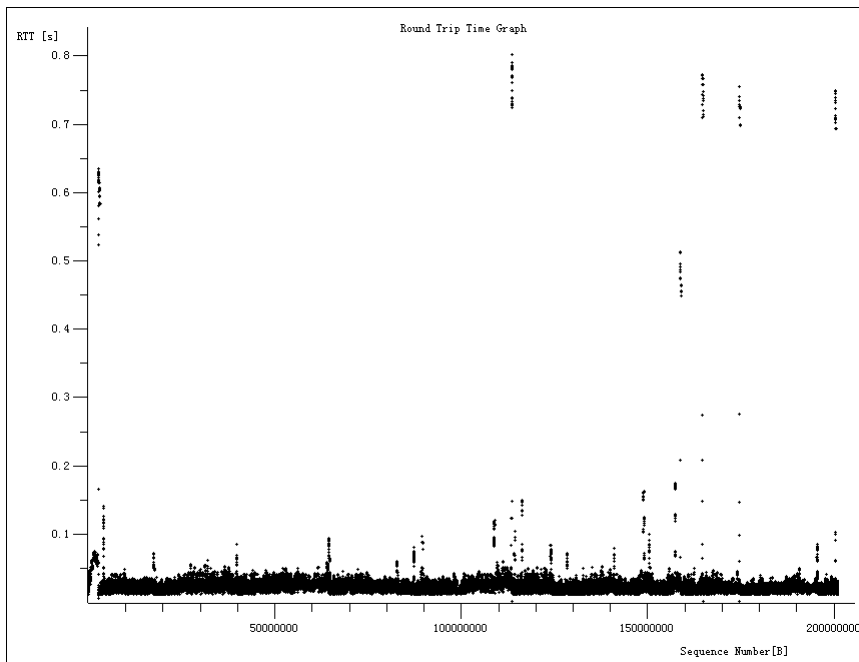
TCP链路的其它一些基本信息，如WS、Time stamping、SACK、MTU等可以在TCP建链的SYN包中的选项字段中查看，如下图。一般情况下MTU=MSS+40，此图没有设置Time stamping，如果设置了也会在选项中显示。



6.2.4 RTT 环回时延

Wireshark统计的RTT时延是发出某个包到收到它的ACK之间的时间差，因此也**必须使用发送端的抓包文件**，使用接收端抓包文件得到的仅仅是接收端的包处理时间。

选中一个**数据包（不能是ACK）**，点击菜单Statistics -> TCP Stream Graph -> Round Trip Time Graph，即可生成RTT时序图，从RTT时序图可以很方便的观察RTT环回时延。

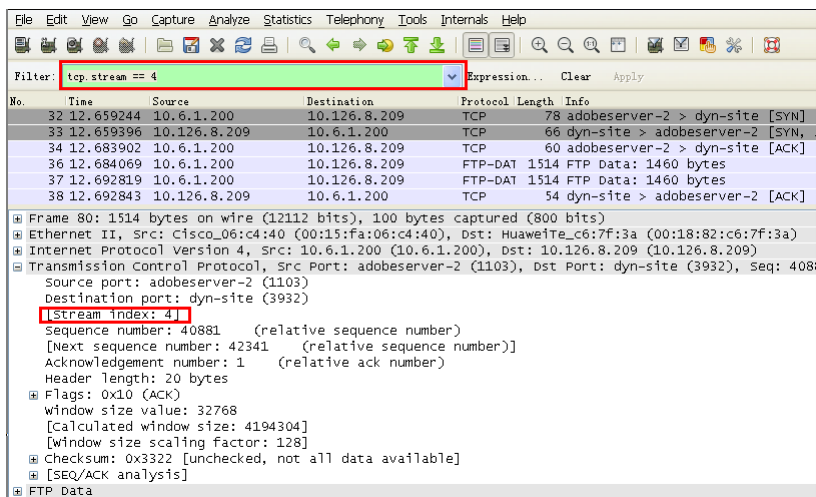


点击RTT时延图上的某个点会直接跳转到Wireshark包列表区对应的包。

RTT时延图主要用于找出某此传输过程中时延很大的点，或者端到端链路中传输时延较大的某段。

6.2.5 如何过滤某一条 TCP 链路

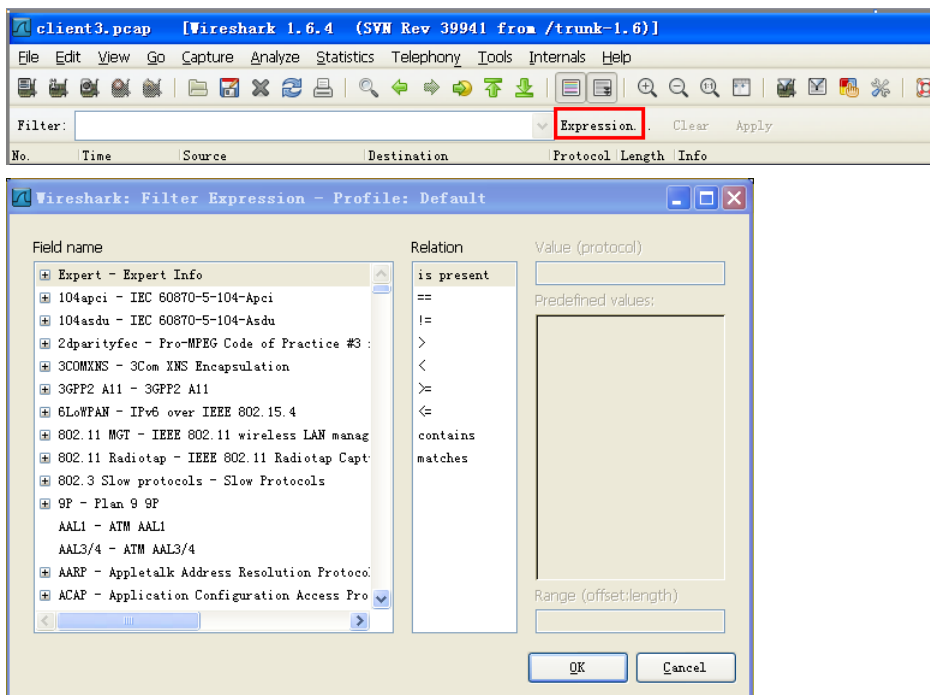
一对TCP连接由源、目的IP，源、目的端口号唯一确定，据此Wireshark提供了过滤某一条TCP链路的功能。**选中某条TCP链路中的一个包**，点击菜单Analyze -> Follow TCP Stream即可过滤出这一条TCP链路。在之前的版本中，这个功能等同于在显示过滤器中设置该链路的源、目的IP，源、目的端口号进行过滤，新版的Wireshark使用了标识Stream Index来代表这个组合，因此也可以在显示过滤器栏直接输入tcp.stream == x来过滤某条TCP链接。如图：



6.3 显示滤波器分析法

6.3.1 显示滤波器表达式

Wireshark显示滤波器的基本表达式为：字段 关系 值，同时可以使用连接符将各个表达式组合起来。



Wireshark能解出的所有字段都可以作为表达式的左端。比较特殊的是协议字段，如果只输入协议字段，则表示“存在”，包含此协议的数据包会被滤出。

关系：如上图所示，共有9种关系表达式，其中6种符号依图中从上到下的顺序又可以表示为eq、ne、gt、lt、ge、le。

值：主要涉及IP、MAC地址的写法，进制等，可以按F1参考帮助文件。

连接符：

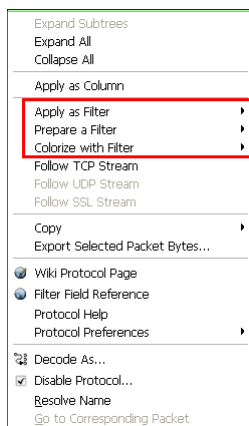
- 逻辑与：and 或者 &&
- 逻辑或：or 或者 ||
- 逻辑异或：xor 或者 ^
- 逻辑非：not 或者 !
- 优先级：()
- 子串操作符：[] 比如想比较一个MAC地址的前两个字节，可以使用如下的表达式

```
eth.src[2] == 02:50
```

Wireshark会自动对显示滤波器的合法性进行检查，符合语法规则的写法会显示绿色，不符合语法规则的表达式会以红色提示。

6.3.2 如何把抓包中的某个字段设为滤波器

在协议解析区找到需要作为滤波器的字段，右键，有三个选项可以方便的把抓包中的某个字段设为滤波器。



Apply as Filter: 直接将选择的字段作为滤波器并执行；

Prepare a Filter: 将选择的字段准备为滤波器，将表达式显示出来，但不执行，可以继续编辑；

Colorize with Filter: 不过滤，只是将符合规则的数据包以不同的颜色标注出来；

6.3.3 常用显示滤波器

常用的显示滤波器如下：

eth.src; eth.dst: 按MAC源、目的地址过滤；

ip.src; ip.dst: 按IP源、目的地址过滤；

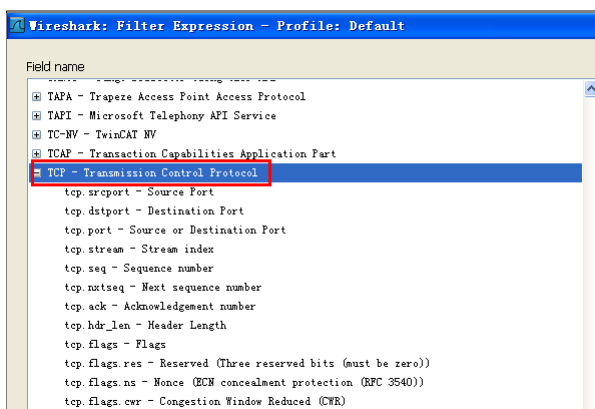
ip.len: 按IP包长过滤，即按MTU过滤；

ip.id: 按IP序号过滤；

涉及TCP的常用滤波器在下一节专门叙述，这里不列出。

6.3.4 TCP 显示滤波器族

涉及TCP的显示滤波器在Filter Expression中可以找到全集，数目较多，这里我们描述在LTE数传问题分析中常用的一些滤波器。



tcp.stream: 唯一标识一条TCP链路，相当于TCP源、目的IP+源、目的端口；

tcp.seq: TCP序号；

tcp.ack: 接收端ACK确认的TCP包序号；

tcp.analysis.lost_segment: 丢包，Wireshark只是判断当前包序号与前一个包序号，如果当前包序号大于前一个包序号且不连续，则判断为lost segment，不一定是实际意义上的丢包，需要进一步判断；

229	8.068957	10.26.192.18	89.9.160.46	TCP	stvp > complex-link [ACK] Seq=10915 Ack=1 win=522720 Len=1210
230	8.068972	10.26.192.18	89.9.160.46	TCP	[TCP previous segment lost] stvp > complex-link [ACK] Seq=13335 Ack=1 win=522720 Len=1210

tcp.analysis.fast_retransmission: 快速重传包，接收端在收到该重传包之前恰好收到了Max. Duplicate ACKs个对该重传包的DUP ACK。

6870 13.308573	80.32.2.202	2.8.11.4	TCP	[TCP Dup ACK 6867#1] ftp-data > neod2 [ACK] Seq=1 Ack=5698571
6871 13.308587	80.32.2.202	2.8.11.4	TCP	[TCP Dup ACK 6867#2] ftp-data > neod2 [ACK] Seq=1 Ack=5698571
6872 13.308604	2.8.11.4	80.32.2.202	FTP-DATA	[TCP Fast Retransmission] FTP Data: 1360 bytes

tcp.analysis.retransmission: 重传包:

1271 11.072195	10.10.1.3	109.47.253.224	FTP-DATA	[TCP Retransmission] FTP Data: 1280 bytes
1272 11.072201	10.10.1.3	109.47.253.224	FTP-DATA	[TCP Retransmission] FTP Data: 1280 bytes
1273 11.072207	109.47.253.224	10.10.1.3	TCP	optima-vnet > ftp-data [ACK] Seq=1 Ack=70
1274 11.073081	10.10.1.3	109.47.253.224	FTP-DATA	[TCP Retransmission] FTP Data: 1280 bytes

tcp.analysis.duplicate_ack: 重复ACK, 接收端按序接收时, 收到的包序号不是期望的包序号, 则每收到一个非期望的数据包发送一个DUP ACK, 其中ACK位填期望的包序号。

1640 8.213394	89.9.160.46	10.26.192.18	TCP	[TCP Dup ACK 1638#1] complex-link > stvp [ACK] Seq=1 Ack=1108385 win=524288 Len=0
1642 8.213418	89.9.160.46	10.26.192.18	TCP	[TCP Dup ACK 1638#2] complex-link > stvp [ACK] Seq=1 Ack=1108385 win=524288 Len=0
1644 8.213441	89.9.160.46	10.26.192.18	TCP	[TCP Dup ACK 1638#3] complex-link > stvp [ACK] Seq=1 Ack=1108385 win=524288 Len=0

tcp.analysis.out_of_order: 乱序包, Wireshark判断当前包序号小于前一个收到的包序号则认为发生了乱序:

2778 8.928851	10.26.192.18	89.9.160.46	TCP	stvp > complex-link [ACK] Seq=1632931 Ack=1 win=522720 Len=1210
2779 8.928866	89.9.160.46	10.26.192.18	TCP	complex-link > stvp [ACK] Seq=1 Ack=1634141 win=524288 Len=0
2780 8.929882	10.26.192.18	89.9.160.46	TCP	[TCP Out-of-order] stvp > complex-link [ACK] Seq=1631721 Ack=1

tcp.analysis.bytes_in_flight: 发送端已经发出但是尚未收到确认的数据量:

tcp.checksum.bad: CRC校验出错的TCP包:

806 0.399675	192.168.2.254	192.168.2.64	TCP	seagull-ats > hbc1 [ACK] Seq=1 Ack=3185 win=32768 [TCP CHECKSUM INCORRECT] Len=0
--------------	---------------	--------------	-----	--

Checksum: 0x86a9 [incorrect, should be 0x0082 (maybe caused by "TCP checksum offload"?)]

[Good Checksum: False]

[Bad Checksum: True]

[Expert Info (Error/Checksum): Bad checksum]

[Message: Bad checksum]

[Severity level: Error]

[Group: Checksum]

tcp.analysis.window_full: TCP发送窗口满:

16616 3.208983	10.144.7.242	10.143.64.17	FTP-DAT	1414 FTP Data: 1360 bytes
16617 3.208988	10.144.7.242	10.143.64.17	FTP-DAT	1414 FTP Data: 1360 bytes
16618 3.208995	10.144.7.242	10.143.64.17	FTP-DAT	1078 [TCP Window Full] FTP Data: 1024 bytes
16619 3.211644	10.143.64.17	10.144.7.242	TCP	60 cspmlockmgr > ftp-data [ACK] Seq=1 Ack=14682785 win=524288 Len=0
16620 3.211937	10.143.64.17	10.144.7.242	TCP	60 cspmlockmgr > ftp-data [ACK] Seq=1 Ack=14685505 win=524288 Len=0

tcp.analysis.window_update: 一个ACK并不确认任何新数据, 只是用来增加窗口的右边沿, 被称为窗口更新。通常发生在接收窗口收缩之后用于窗口大小的恢复。

996 0.000906	2.10.5.30	72.30.90.100	TCP	mc3ss > ftp-data [ACK] Seq=4233007290 Ack=1570824525 win=32725 Len=0
997 0.000017	2.10.5.30	72.30.90.100	TCP	[TCP Window Update] mc3ss > ftp-data [ACK] Seq=4233007290 Ack=1570824525 win=32810 Len=0

tcp.analysis.zero_window: 窗口的左边沿到达右边沿, 则称其为一个零窗口, 此时发送方不能够发送任何数据, 即我们常说的窗口收缩到0。

44678 0.000025	192.168.81.16	90.32.1.60	TCP	atex_e1md > hermes [ACK] Seq=2924484365 Ack=3455591310 win=21 Len=0
44679 0.000905	192.168.81.16	90.32.1.60	TCP	[TCP ZeroWindow] atex_e1md > hermes [ACK] Seq=2924484365 Ack=3455594030 win=0 Len=0

6.3.5 利用显示过滤器分析丢包

Wireshark 对应判断丢包的显示过滤器是 `tcp.analysis.lost_segment`，但是使用 `tcp.analysis.lost_segment` 来判断丢包有一个缺点，因为 `tcp.analysis.lost_segment` 只是判断前后包的序号是否连续，如果只是乱序一或者两个包 Wireshark 也会判定为 `tcp.analysis.lost_segment`，但实际上此时对链路是没有影响的。

这里推荐使用 `tcp.analysis.retransmission` 来分析 TCP 的丢包乱序问题，不管是丢包还是乱序，只要造成了重传，都会影响 TCP 的发送窗口从而影响吞吐率，使用 `tcp.analysis.retransmission` 能过滤这一类问题。

以三点抓包为例来说明分析过程，假设 TCP 数据的流向是 $A \rightarrow B \rightarrow C$ ，C 点是客户端 PC，C 点的抓包使用 `tcp.analysis.retransmission` 过滤发现有重传，那么我们按照 6.3.2 介绍的方法将 `tcp.seq == uvwxyz` 设置为过滤器，拷贝这个过滤器 `tcp.seq == uvwxyz` 到 A、B、C 三点的抓包文件中同时执行，根据过滤得到的包数量可以判断错误类型。如下表，`tcp.seq == uvwxyz` 在 A、B、C 三点过滤得到的包数量 vs TCP 异常类型表格：

A点	B点	C点	TCP异常类型
0	0	1	A或者B点漏抓包
2	1	1	AB点之间丢包
2	2	1	BC点之间丢包
2	2	2	乱序或者超时重传
3	2	2	超时重传

这里只列出了最一般的情况，实际情况中镜像口漏抓包、TPE 算法打开等都会影响各点的包数，出现其它各种组合，这里未一一列出。

6.3.6 利用显示过滤器分析乱序

少量的乱序不会对吞吐量造成影响，只有当乱序的数量大于等于系统配置的 `Max. Duplicate ACKs` 时，触发接收端发送的 `DUP ACK` 数量达到门限导致发送端误判为丢包而引起快速重传，对吞吐量造成影响。

因此，乱序的问题可以分为两类来看待：未造成重传的少量乱序和引起重传的乱序。

多个点都发生少量乱序合在一起可能使乱序的数量大于等于系统配置的 `Max.`

Duplicate ACKs引起重传，所以有时候需要分析某点抓包的少量乱序，这种情况可以使用显示滤波器tcp.analysis.out_of_order来过滤，如图：

195 4.912098	77.88.99.74	89.9.162.232	HTTP	1314 Continuation or non-HTTP traffic[Packet size limited during capture]
196 4.912109	77.88.99.74	89.9.162.232	HTTP	1314 [TCP Previous segment lost] Continuation or non-HTTP traffic[Packet s
197 4.912121	89.9.162.232	77.88.99.74	TCP	66 molly > http [ACK] Seq=447 Ack=34021 win=524288 Len=0 SLE=35281 SRE=3
198 4.912130	77.88.99.74	89.9.162.232	HTTP	1314 [TCP out-of-order] Continuation or non-HTTP traffic[Packet size limit
199 4.912150	89.9.162.232	77.88.99.74	TCP	54 molly > http [ACK] Seq=447 Ack=36541 win=524288 Len=0

引起重传的乱序可以使用tcp.analysis.retransmission来过滤，分析方法同上一节。

6.3.7 利用显示滤波器分析窗口收缩

TCP接收窗口大小是TCP连接中用于存储输入数据的接收主机的内存缓冲器中字节量，对应于接收端允许发送端发送的剩余数据量，它等于最大接收窗口大小与已接收和确认但尚未被应用程序检索的数据量之间的差值。

接收窗口大小主要受应用程序检索接收窗口中累积数据的速度影响，如果应用程序不能检索数据，接收窗口就可以开始填充，导致接收端通告了更小的当前窗口大小。在极个别的情况下，整个最大接收窗口都会填满，导致接收端通告了 0 字节的窗口大小。在这种情况下，发送端必须停止发送数据，直到接收窗口已经清除为止。因此，接收窗口的大小是影响TCP速率的重要因素。

接收窗口收缩到0的危害最大，这种情况我们可以使用显示滤波器tcp.analysis.zero_window来过滤0窗口。但是，窗口收缩的场景并非总能收缩到0，有时候窗口会呈收缩的趋势但并未收缩到0又开始恢复，这个时候我们可以使用显示滤波器tcp.window_size < xxxxx（xxxxx为某个定值，比如1/2最大接收窗口大小）来过滤这样的包。

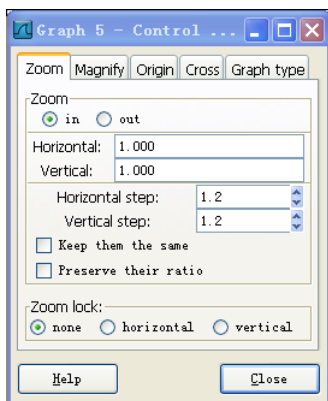
593186 146.357603	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565208305 Win=20400 Len=0
593187 146.357603	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565208305 Win=20400 Len=0
593188 146.357645	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565213745 Win=14960 Len=0
593189 146.363793	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565216465 Win=12240 Len=0
593190 146.363820	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565219185 Win=9520 Len=0
593191 146.363880	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565221905 Win=6800 Len=0
593192 146.363886	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565224625 Win=4080 Len=0
593193 146.363911	10.102.13.140	10.102.16.117	TCP	2154 > ftp-data [ACK] Seq=1 Ack=565227345 Win=1360 Len=0
593194 146.435931	10.102.16.120	10.102.16.255	NBNS	Name query NB RMS.HUAWEI.COM<00>
593195 146.447841	Cisco_cc:5d:70	Spanning-tree-(for-br	STP	Conf. Root = 4580/00:0c:cf:14:f3:140 Cost = 3015 Port = 0x2051
593196 146.451826	10.102.13.140	10.102.16.117	TCP	[TCP ZeroWindowProbe] 2154 > ftp-data [ACK] Seq=1 Ack=565228705 Win=0 Len=0
593197 146.932035	10.102.16.117	10.102.13.140	TCP	[TCP ZeroWindowProbe] ftp-data > 2154 [ACK] Seq=565228705 Ack=1 Win=524288
593198 146.935842	10.102.13.140	10.102.16.117	TCP	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 2154 > ftp-data [ACK] Seq=1 Ack=
593199 147.1186604	10.102.16.120	10.102.16.255	NBNS	Name query NB RMS.HUAWEI.COM<00>
593200 147.934944	10.102.16.120	10.102.16.255	NBNS	Name query NB RMS.HUAWEI.COM<00>
593201 147.943780	10.102.16.117	10.102.13.140	TCP	[TCP ZeroWindowProbe] ftp-data > 2154 [ACK] Seq=565228705 Ack=1 Win=524288
593202 147.975805	10.102.13.140	10.102.16.117	TCP	[TCP ZeroWindowProbeAck] [TCP ZeroWindow] 2154 > ftp-data [ACK] Seq=1 Ack=
593203 148.435630	Cisco_cc:5d:70	Spanning-tree-(for-br	STP	Conf. Root = 4580/00:0c:cf:14:f3:140 Cost = 3015 Port = 0x2051
593204 148.757465	10.102.16.120	10.102.16.255	NBNS	Name query NB RMS.HUAWEI.COM<00>
593205 149.395756	10.102.13.140	10.102.16.117	TCP	[TCP Window Update] 2154 > ftp-data [ACK] Seq=1 Ack=565228705 Win=524288 L
593206 149.395816	10.102.16.117	10.102.13.140	FTP-DATA	FTP Data: 1360 bytes

窗口收缩一般与PC性能有关，如果PC性能较差或者计算机同时运行着多个应用程序占用了大量资源，就会导致操作系统来不及处理TCP接收窗缓存中的数据，导致接收窗逐渐填满，窗口越来越小。在CPU占用率不高的情况下，一般通过更换性能更高的PC机来解决这个问题。

6.4 图形分析法

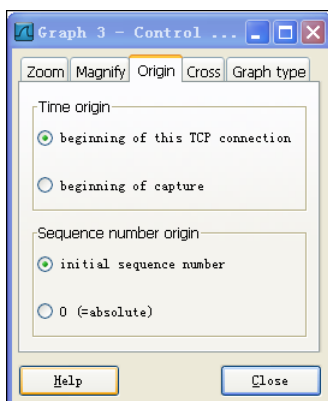
6.4.1 Wireshark 的图形控制

当使用Wireshark菜单Statistics -> TCP Stream Graph生成各类图形时，一个图形控制对话框会同时自动生成。如下图：



- Zoom页签：

- Horizontal和Vertical两栏显示了当前图形水平和垂直方向的放大倍数；
- Horizontal step和Vertical step两栏设定当你放大图形时的步长因子；
- Keep them the same: 保持水平和垂直方向放大倍数相同；
- Preserve their ratio: 保证水平和垂直方向放大比率相同，比如水平方向放大1.2倍垂直方向放大2倍，锁定比率后，水平方向如果改为2.4倍，则垂直方向自动改变为4倍；
- Zoom lock: 锁定某个方向不进行放大；



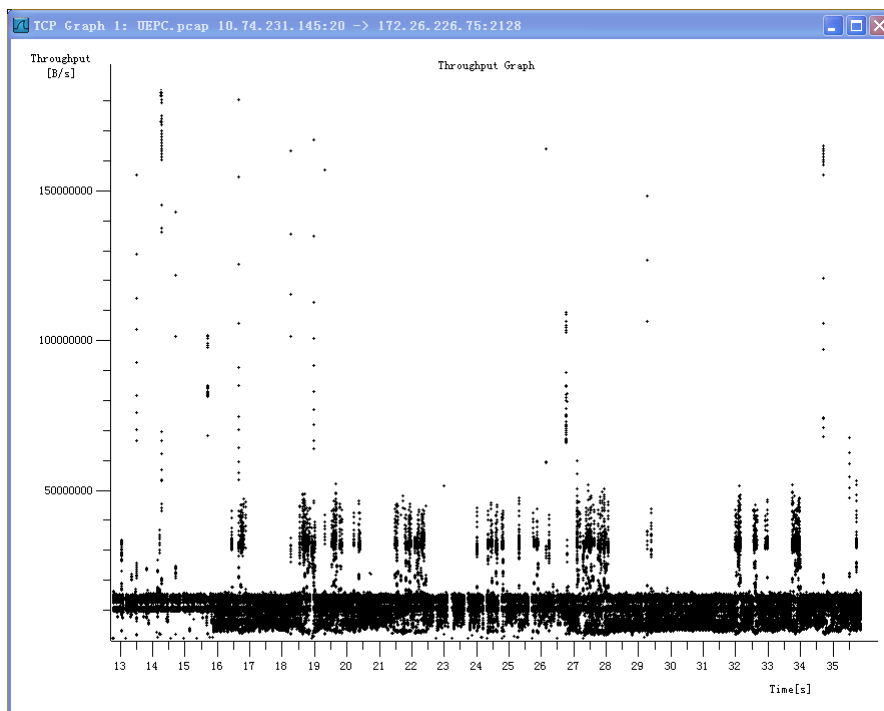
- Magnify页签

设定和局部放大有关的选项。

- Origin页签：
 - Time origin: 设定作图是依据选定的TCP连接还是整个抓包文件;
 - Sequence number origin: 选择使用ISN还是相对序列号;
- Cross页签：
 - Crosshairs: 控制是否打开十字光标;
- 快捷键：
 - 放大: 鼠标中键或者“i”键或者“+”键;
 - 缩小: “o”键或者“-”键;
 - **局部放大: Ctrl+鼠标右键, 此步操作会弹出一个新的图形框。**
 - 拖拽: 鼠标右键;
 - 选中某个包: 鼠标左键;
 - 恢复: “r”键;
 - 放大情况下的移动: 方向键（每次100个像素），shift+方向键（每次10个像素），ctrl+方向键（每次一个像素）;

6.4.2 TCP 流量图 Throughput Graph

选中某个TCP包，点击菜单Statistics -> TCP Stream Graph -> Throughput Graph可以生成TCP流量图，其横轴是时间，单位秒s，纵轴是流量，单位是B/s，注意这里的单位是字节每秒，不是我们在网络上常用的比特每秒，需要做一个折算。



6.4.3 TCP 接收窗口图 Window Scaling Graph

选中某个TCP包，点击菜单Statistics -> TCP Stream Graph -> Window Scaling Graph，即可生成TCP接收窗口图。TCP接收窗口是根据Wireshark如下标识作图的：

```

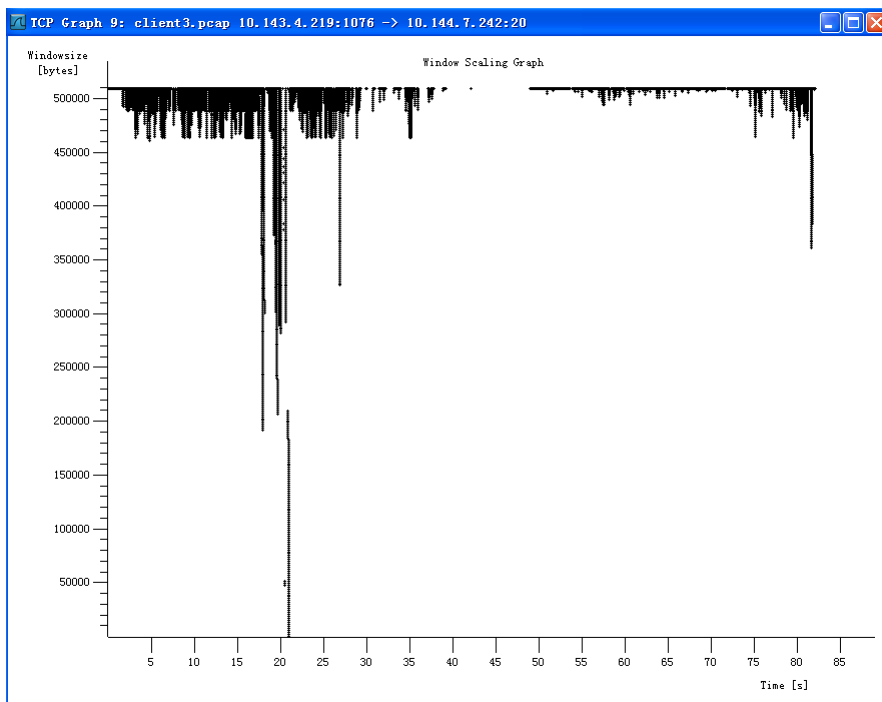
Frame 41: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: HuaweiTe_c6:7f:3a (00:18:82:c6:7f:3a), Dst: Cisco_06:c4:40 (00:15:fa:06:c4:40)
Internet Protocol Version 4, Src: 10.126.8.209 (10.126.8.209), Dst: 10.6.1.200 (10.6.1.200)
Transmission Control Protocol, Src Port: dyn-site (3932), Dst Port: adobeserver-2 (1103), Seq: 1, Ack: 5841, Len: 0
Source port: dyn-site (3932)
Destination port: adobeserver-2 (1103)
[Stream index: 4]
Sequence number: 1 (relative sequence number)
Acknowledgement number: 5841 (relative ack number)
Header length: 20 bytes
Flags: 0x10 (ACK)
window size value: 32768
[Calculated window size: 524288]
[window size scaling factor: 16]
Checksum: 0x1f37 [validation disabled]
[SEQ/ACK analysis]
    
```

要注意的是，同一个tcp stream index下由于有数据和ACK分别从两个方向发送（参见6.2.1节），所以选中数据和选中ACK作出的图是不一样的。同时，也必须保证在抓包文件中包含该条链路的建链包SYN，否则作出的图也不是扩展后的接收窗口。

如下图，没有抓到建链包的情况：

```
Frame 10: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: Dell_8c:cc:02 (f0:4d:a2:8c:cc:02), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 172.26.226.33 (172.26.226.33), Dst: 10.74.231.145 (10.74.231.145)
Transmission Control Protocol, Src Port: openmath (1473), Dst Port: ftp-data (20), Seq: 1, Ack: 8761, Len: 0
  Source port: openmath (1473)
  Destination port: ftp-data (20)
  [Stream index: 0]
  Sequence number: 1 (relative sequence number)
  Acknowledgement number: 8761 (relative ack number)
  Header length: 20 bytes
  Flags: 0x10 (ACK)
  Window size value: 32768
  [Calculated window size: 32768]
  [window size scaling factor: -1 (unknown)]
  Checksum: 0x8032 [validation disabled]
  [SEQ/ACK analysis]
```

下图是一次TCP传输过程中的接收窗口变化：



6.4.4 时序图介绍 Time-Sequence Graph

Wireshark的时序图产生一个TCP序号vs.时间的图形。在Wireshark的菜单Statistics -> TCP Stream Graph -> Time-Sequence Graph下,有两种格式的时序图,steven和tcptrace。

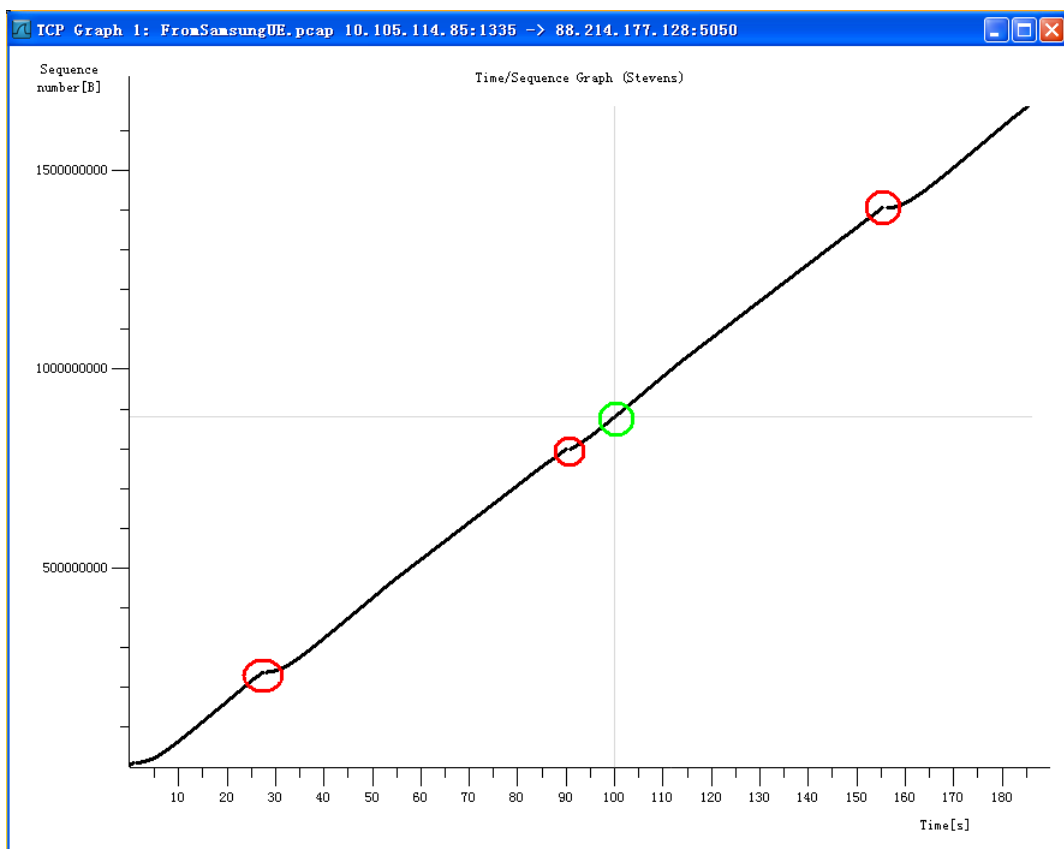
Stevens时序图描述了流量（序号）随着时间的变化，主要用于查看流量变化的情况，是否有中断、丢包、大时延的情况，命名取自W. Richard Stevens写的经典书籍《TCP/IP协议详解》。

Tcptrace时序图除了stevens时序图的内容外，还记录了对端ACK值和接收窗口的变化，内容较stevens时序图要丰富。Tcptrace原是Ohio University的Shawn Ostermann写的一个工具。

6.4.5 Stevens 时序图 Time-Sequence Graph (Stevens)

正常情况下，如果TCP速率稳定，那么在stevens时序图上看到的将是一条笔直上升的斜线，它的斜率等于速率。

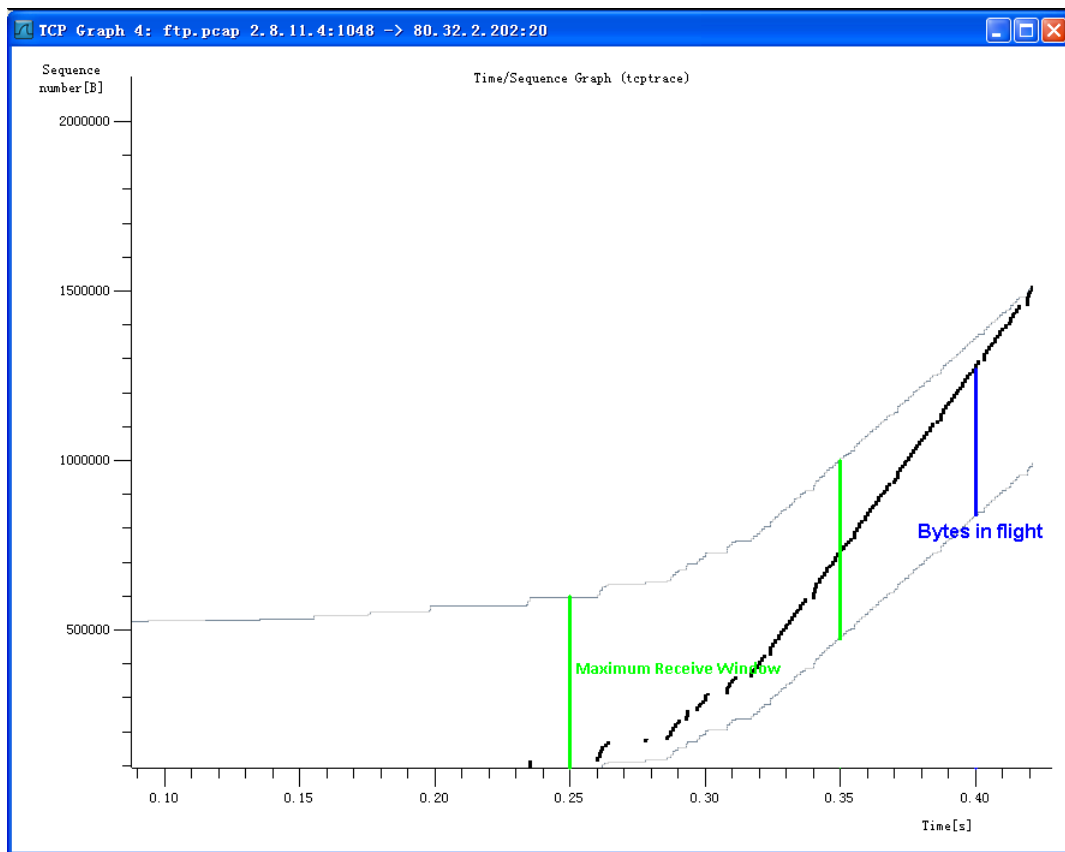
如下图，是p国某局点使用三星终端进行下载抓包得到的stevens时序图，从图上可以看到，虽然斜线有三个不连续的点，但是斜率基本稳定，可以估算下载的速率约在 $900\text{MB}/100\text{s}=72\text{mbps}$ 附近，三个不连续的点暗示了TCP传输在这三个位置发生了重传，但速率很快恢复。



stevens时序图非常简洁，一眼就能看出问题所在。

6.4.6 tcptrace 时序图 Time-Sequence Graph (tcptrace)

相比Stevens时序图，tcptrace时序图增加了对端ACK值和接收窗口变化的序号。如下图，中间黑色的粗线代表了发送的包，下方浅色的线代表了上一个ACK确认的包序号，上方浅色的线表征了TCP窗口，它等于上一个TCP ACK序号再加上TCP链路的window size。



由于tcptrace时序图相当于增加了变量，我们可以获取更为丰富的TCP链路变化信息。

6.4.7 利用 tcptrace 时序图分析 TCP 链路异常

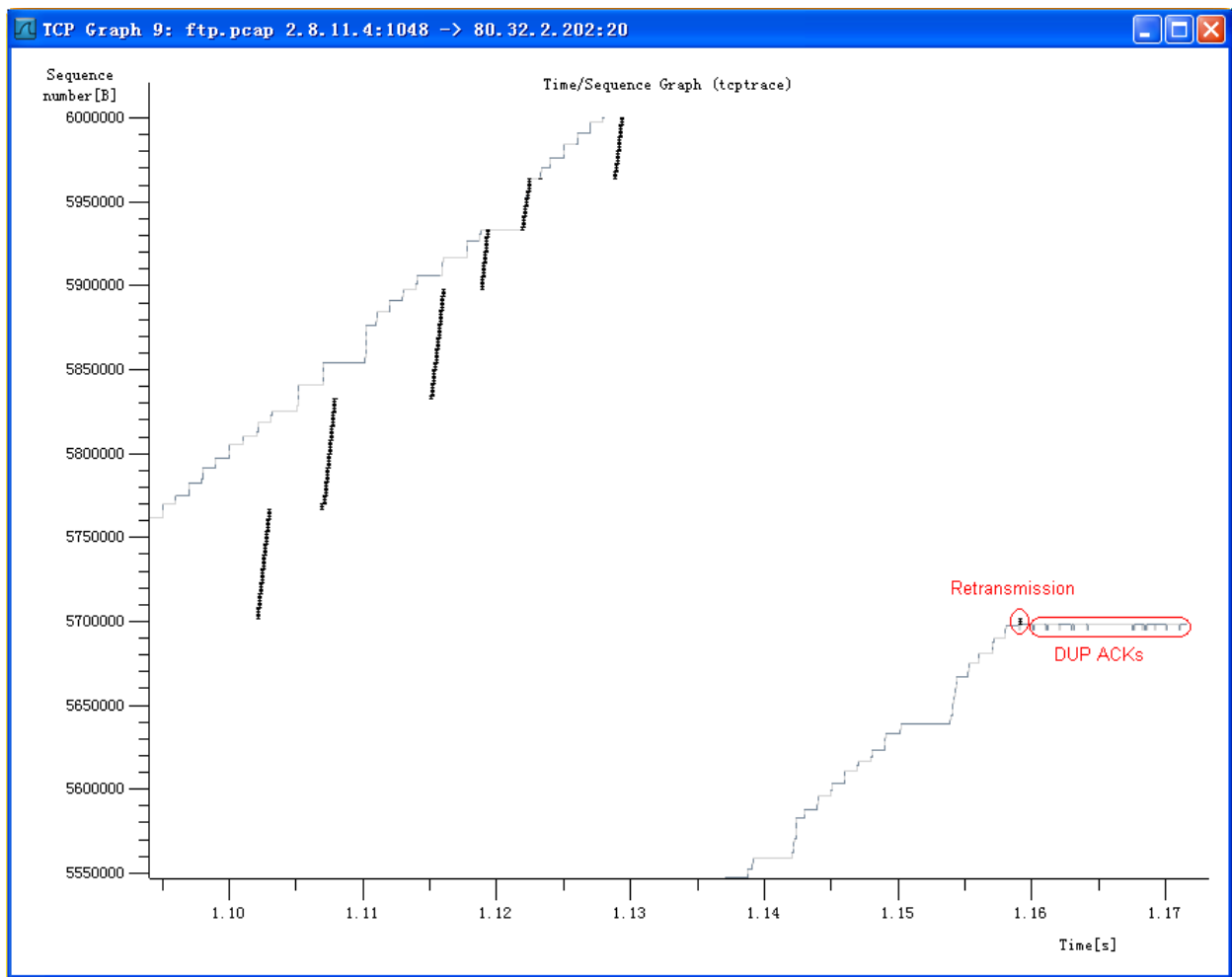
正常的tcptrace时序图随着时间的增长，代表发送包序号的黑色线段和下方代表ACK确认包序号的浅色线段都应该向上增长，如果tcptrace时序图中出现了任何线段的中断、重复、离散的点等等情况，则可能TCP链路出现了异常，需要深入分析。

直接点击tcp时序图中出问题的点在Wireshark包列表区中会直接跳转到对应的包。

如下图，远离黑色线段主体的一小段黑色线段是重传包，而在这之后，还不断收到对这个包的重复确认，显然，这是一个发送端的抓包文件。

NOTE

注意，*DUP ACK* 和很久没有收到 *ACK* 是不同的，很长时间没有收到 *ACK* 是一条笔直的直线，而连续的 *DUP ACK* 是类似 “7” 号这样的符号连接在一起。



7 使用 Wireshark 分析 LTE 数传问题流程

7.1.1 常规步骤

- 首先判断抓包文件是否是分段的多个抓包文件需要合并，先进行多文件合并操作，参见[5.2.2多文件合并](#)。
- 观察抓包文件中是否抓到有“杂包”，过滤掉其它无用的TCP流（参见[6.2.5如何过滤某一条TCP链路](#)），保存成新的抓包文件，再重新打开（参见[5.2.1仅保存需要的数据](#)）。
- 打开合并后的抓包，初步判断一下抓包文件是否抓到了包头（决定采用相对序号还是绝对序号），是否有分片（决定是否设置重组），是否需要进行ESP解密（决定是否设置解密）等，进行相关设置。参见[4.2参数设置](#)。
- 使用IO Graphs观察一下流量的变化情况，确认是否存在问题。参见[6.1.3流量统计](#)

[IO Graphs](#)。

- E. 使用Expert Info Composite观察一下抓包文件中包含的异常信息，初步判断一下问题类型。参见[6.1.4汇总所有日志的专家信息 Expert Info Composite](#)。
- F. 根据初步判断的问题类型和分析人员的使用习惯选择合适的分析方法（[6.3显示滤波器分析法](#)或者[6.4图形分析法](#)）进行分析。
- G. 得出结论。

7.1.2 使用 Wireshark 分析 LTE 数传问题流程图

