# **Wireshark基本用法**

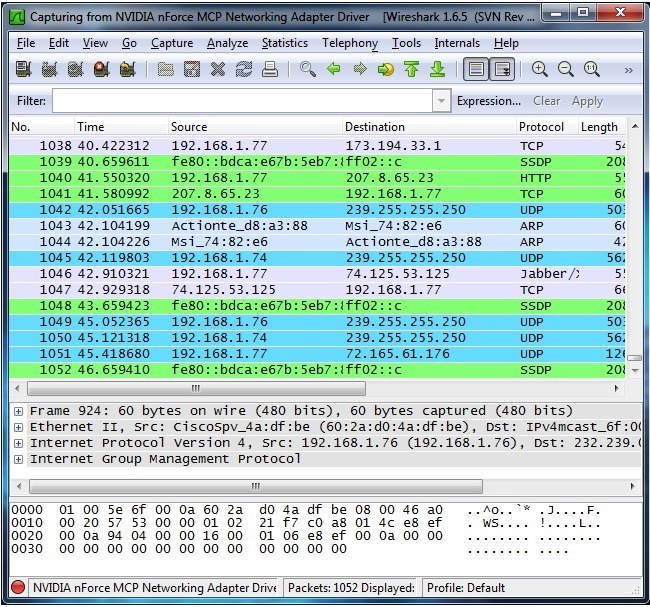
**抓取报文:**

下载和安装好Wireshark之后，启动Wireshark并且在接口列表中选择接口名，然后开始在此接口上抓包。例如，如果想要在无线网络上抓取流量，点击无线接口。

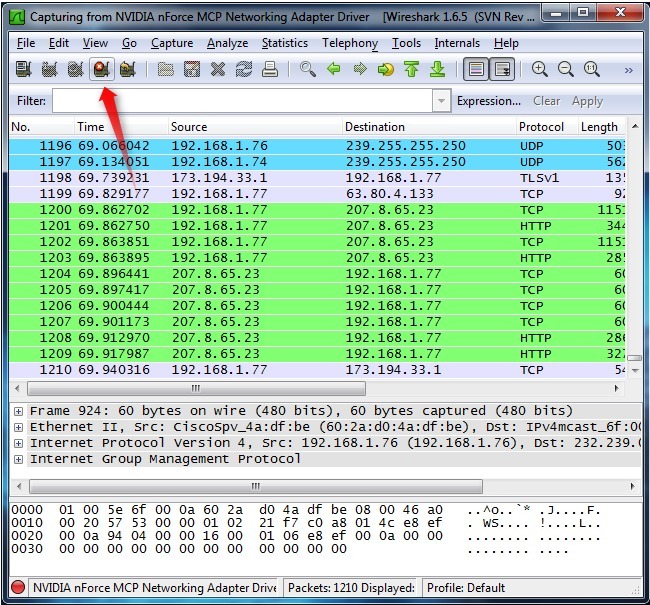
点击Capture Options可以配置高级属性，但现在无此必要。

　　点击接口名称之后，就可以看到实时接收的报文。如果抓取的接口是无线并且选项选取的是混合模式，那么也会看到网络上其他报文。

　　默认显示报文接收时间（相对开始抓取的时间点），源和目标IP地址，使用协议和报文相关信息。点击某一行可以在下面两个窗口看到更多信息。“+”图标显示报文里面每一层的详细信息。底端窗口同时以十六进制和ASCII码的方式列出报文内容。

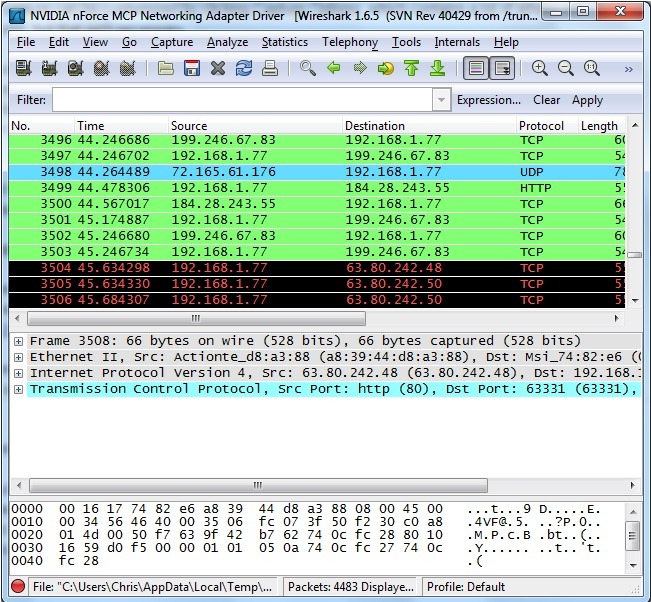


　　需要停止抓取报文的时候，点击左上角的停止按键。



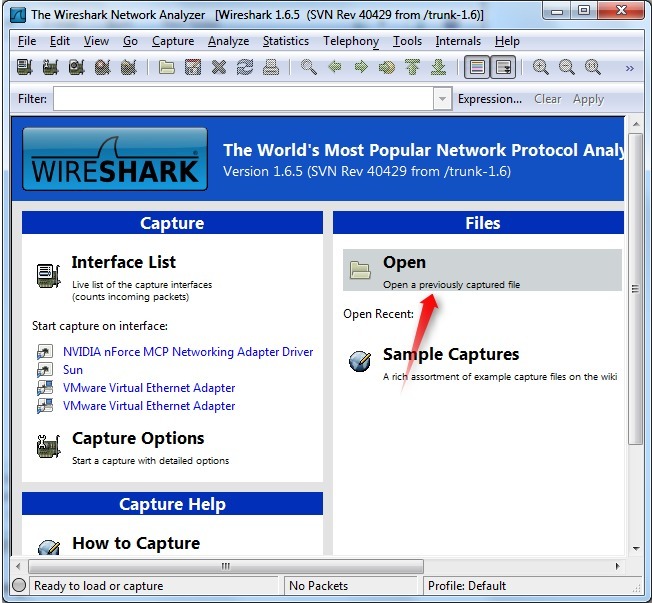
**色彩标识:**

　　Wireshark通过颜色让各种流量的报文一目了然。比如默认绿色是TCP报文，深蓝色是DNS，浅蓝是UDP，黑色标识出有问题的TCP报文——比如乱序报文。



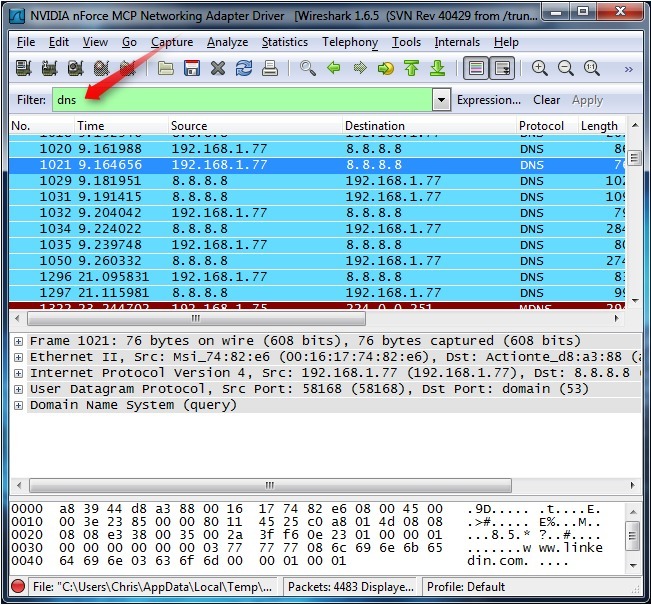
**报文样本:**

　　打开一个抓取文件相当简单，在主界面上点击Open并浏览文件即可。也可以在Wireshark里保存自己的抓包文件并稍后打开。

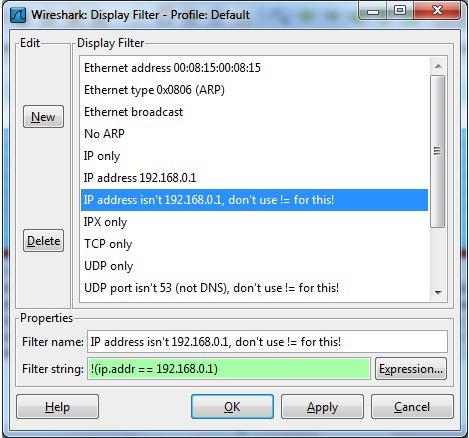


**过滤报文:**

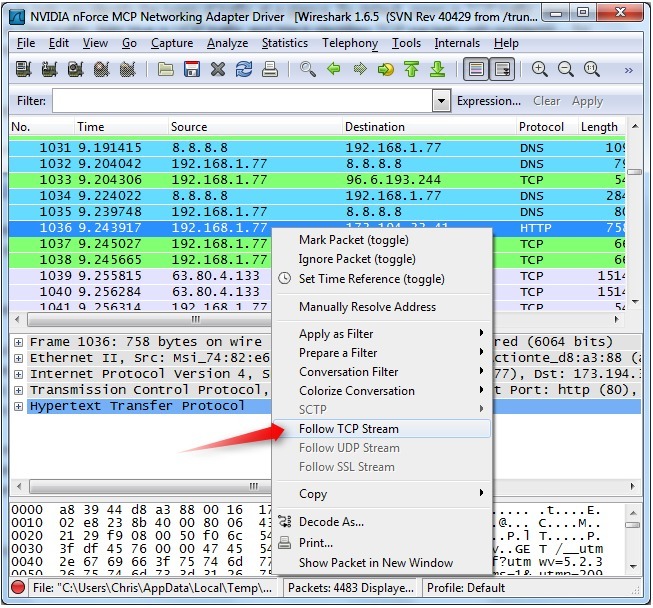
　　最基本的方式就是在窗口顶端过滤栏输入并点击Apply（或按下回车）。例如，输入“dns”就会只看到DNS报文。输入的时候，Wireshark会帮助自动完成过滤条件。



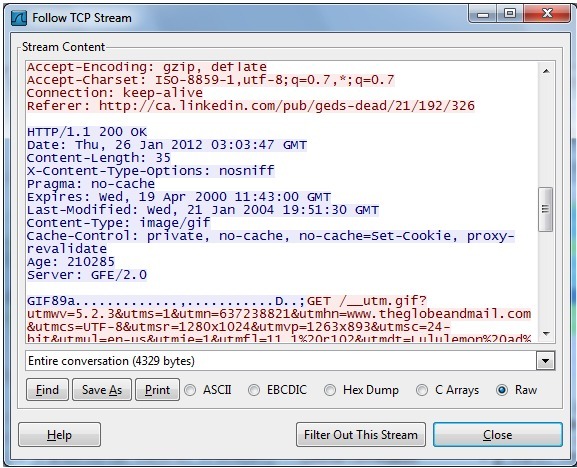
　　也可以点击Analyze菜单并选择Display Filters来创建新的过滤条件。



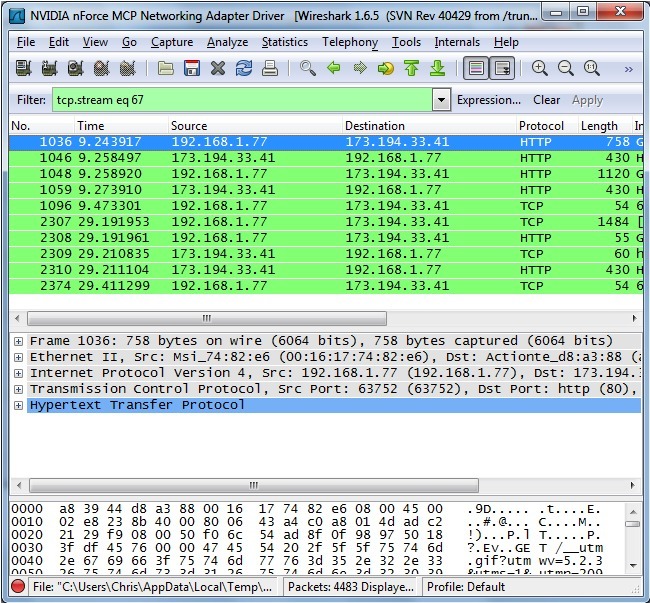
　　另一件很有趣的事情是你可以右键报文并选择Follow TCP Stream。



　　你会看到在服务器和目标端之间的全部会话。



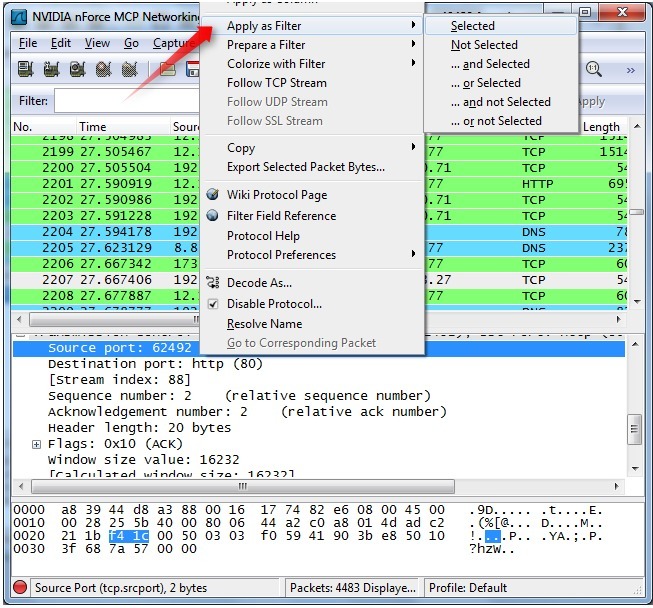
　　关闭窗口之后，你会发现过滤条件自动被引用了——Wireshark显示构成会话的报文。



**检查报文:**

　　选中一个报文之后，就可以深入挖掘它的内容了。

　　也可以在这里创建过滤条件——只需右键细节并使用Apply as Filter子菜单，就可以根据此细节创建过滤条件。



Wireshark是一个非常之强大的工具，第一节只介绍它的最基本用法。

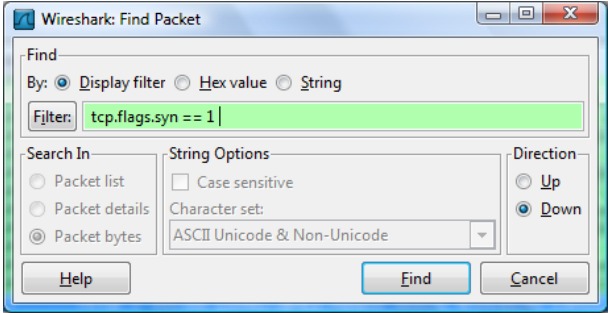
# **一站式学习Wireshark（二）：应用Wireshark观察基本网络协议**

**TCP:**

　　TCP/IP通过三次握手建立一个连接。这一过程中的三种报文是：SYN，SYN/ACK，ACK。

　　第一步是找到PC发送到网络服务器的第一个SYN报文，这标识了TCP三次握手的开始。

　　如果你找不到第一个SYN报文，选择**Edit -> Find Packet**菜单选项。选择Display Filter，输入过滤条件：tcp.flags，这时会看到一个flag列表用于选择。选择合适的flag，tcp.flags.syn并且加上==1。点击Find，之后trace中的第一个SYN报文就会高亮出来了。

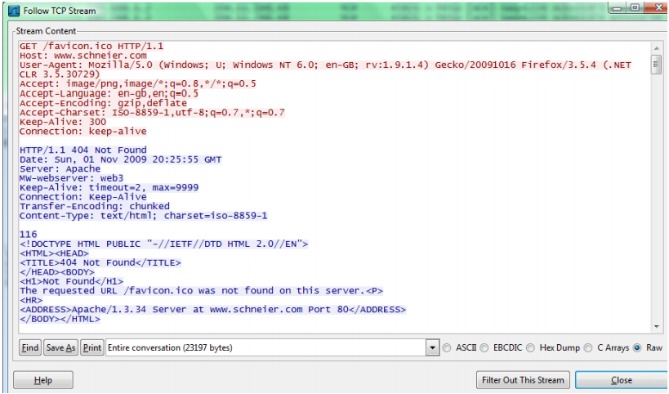


　　注意：Find Packet也可以用于搜索十六进制字符，比如恶意软件信号，或搜索字符串，比如抓包文件中的协议命令。

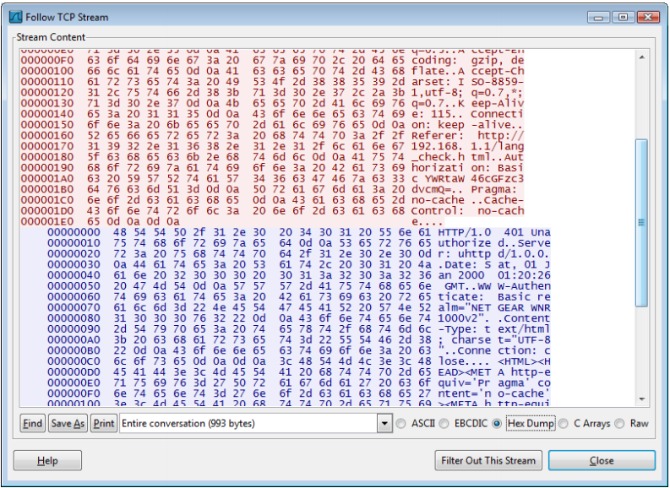
　　一个快速过滤TCP报文流的方式是在**Packet List Panel**中右键报文，并且选择**Follow TCP Stream**。这就创建了一个只显示TCP会话报文的自动过滤条件。

　　这一步骤会弹出一个会话显示窗口，默认情况下包含TCP会话的ASCII代码，客户端报文用红色表示服务器报文则为蓝色。

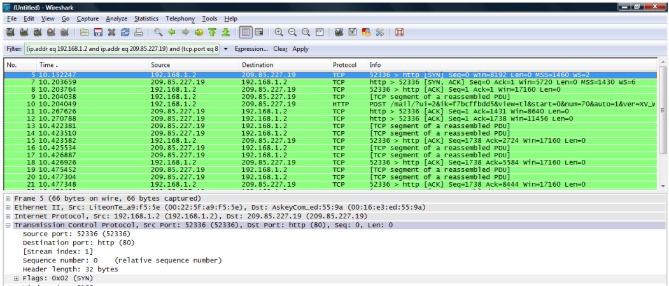
　　窗口类似下图所示，对于读取协议有效载荷非常有帮助，比如HTTP，SMTP，FTP。



　　更改为十六进制Dump模式查看载荷的十六进制代码，如下图所示：



　　关闭弹出窗口，Wireshark就只显示所选TCP报文流。现在可以轻松分辨出3次握手信号。



　　注意：这里Wireshark自动为此TCP会话创建了一个显示过滤。本例中：(ip.addr eq 192.168.1.2 and ip.addr eq 209.85.227.19) and (tcp.port eq 80 and tcp.port eq 52336)

**SYN报文：**

　　图中显示的5号报文是从客户端发送至服务器端的SYN报文，此报文用于与服务器建立同步，确保客户端和服务器端的通信按次序传输。SYN报文的头部有一个32 bit序列号。底端对话框显示了报文一些有用信息如报文类型，序列号。

**SYN/ACK报文：**

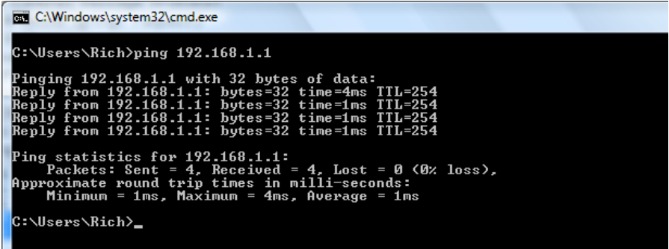
　　7号报文是服务器的响应。一旦服务器接收到客户端的SYN报文，就读取报文的序列号并且使用此编号作为响应，也就是说它告知客户机，服务器接收到了SYN报文，通过对原SYN报文序列号加一并且作为响应编号来实现，之后客户端就知道服务器能够接收通信。

**ACK报文：**

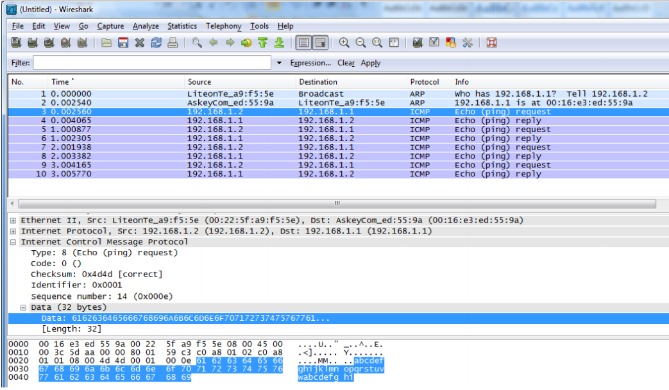
　　8号报文是客户端对服务器发送的确认报文，告诉服务器客户端接收到了SYN/ACK报文，并且与前一步一样客户端也将序列号加一，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

**ARP & ICMP：**

　　开启Wireshark抓包。打开Windows控制台窗口，使用ping命令行工具查看与相邻机器的连接状况。



　　停止抓包之后，Wireshark如下图所示。ARP和ICMP报文相对较难辨认，创建只显示ARP或ICMP的过滤条件。



**ARP报文：**

　　地址解析协议，即ARP（Address Resolution Protocol），是根据[IP地址](http://baike.baidu.com/view/3930.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)获取[物理地址](http://baike.baidu.com/view/883168.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)的一个[TCP/IP协议](http://baike.baidu.com/view/7649.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)。其功能是：[主机](http://baike.baidu.com/view/23880.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)将ARP请求[广播](http://baike.baidu.com/view/35385.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)到网络上的所有主机，并接收返回消息，确定目标[IP地址](http://baike.baidu.com/view/3930.htm" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)的物理地址，同时将IP地址和硬件地址存入本机ARP缓存中，下次请求时直接查询ARP缓存。

最初从PC发出的ARP请求确定IP地址192.168.1.1的MAC地址，并从相邻系统收到ARP回复。ARP请求之后，会看到ICMP报文。

**ICMP报文：**

　　网络控制消息协定（Internet Control Message Protocol，ICMP）用于[TCP/IP](http://zh.wikipedia.org/wiki/TCP/IP" \o "" \t "https://www.cnblogs.com/Jerry-zhao2110/p/_blank)网络中发送控制消息，提供可能发生在通信环境中的各种问题反馈，通过这些信息，令管理者可以对所发生的问题作出诊断，然后采取适当的措施解决。

PC发送echo请求，收到echo回复如上图所示。ping报文被mark成Type 8，回复报文mark成Type 0。

如果多次ping同一系统，在PC上删除ARP cache，使用如下ARP命令之后，会产生一个新的ARP请求。

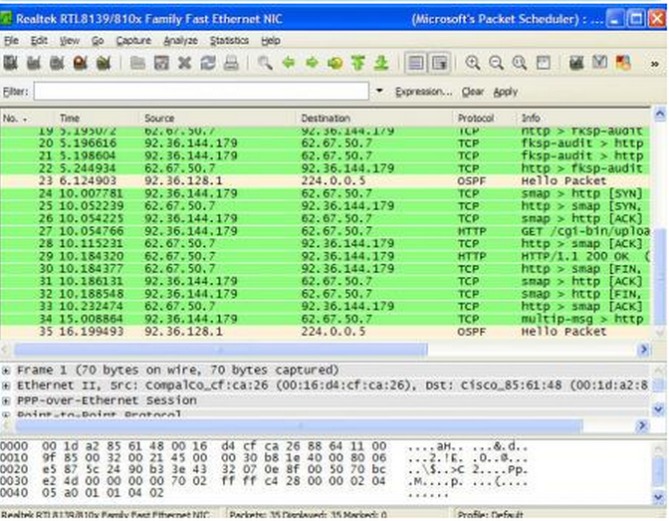
C:\> ping 192.168.1.1

… ping output …

C:\> arp –d \*

**HTTP：**

　　HTTP协议是目前使用最广泛的一种基础协议，这得益于目前很多应用都基于WEB方式，实现容易，软件开发部署也简单，无需额外的客户端，使用浏览器即可使用。这一过程开始于请求服务器传送网络文件。



　　从上图可见报文中包括一个GET命令，当HTTP发送初始GET命令之后，TCP继续数据传输过程，接下来的链接过程中HTTP会从服务器请求数据并使用TCP将数据传回客户端。传送数据之前，服务器通过发送HTTP OK消息告知客户端请求有效。如果服务器没有将目标发送给客户端的许可，将会返回403 Forbidden。如果服务器找不到客户端所请求的目标，会返回404。

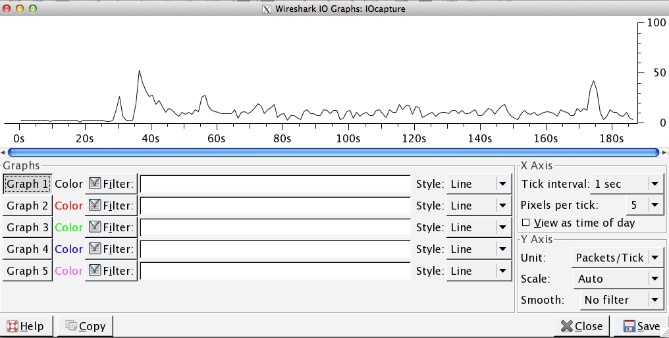
　　如果没有更多数据，连接可被终止，类似于TCP三次握手信号的SYN和ACK报文，这里发送的是FIN和ACK报文。当服务器结束传送数据，就发送FIN/ACK给客户端，此报文表示结束连接。接下来客户端返回ACK报文并且对FIN/ACK中的序列号加1。这就从服务器端终止了通信。要结束这一过程客户端必须重新对服务器端发起这一过程。必须在客户端和服务器端都发起并确认FIN/ACK过程。

# **应用Wireshark IO图形工具分析数据流**

**基本IO Graphs:**

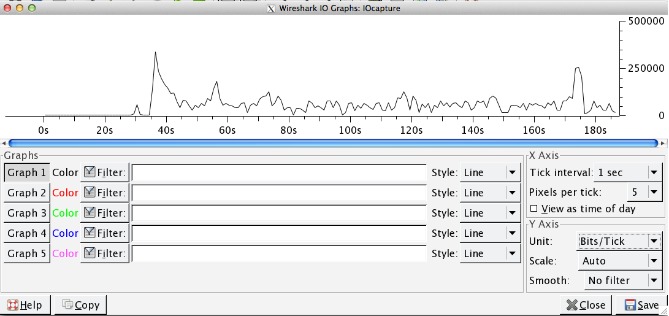
　　IO graphs是一个非常好用的工具。基本的Wireshark IO graph会显示抓包文件中的整体流量情况，通常是以每秒为单位（报文数或字节数）。默认X轴时间间隔是1秒，Y轴是每一时间间隔的报文数。如果想要查看每秒bit数或byte数，点击“Unit”，在“Y Axis”下拉列表中选择想要查看的内容。这是一种基本的应用，对于查看流量中的波峰/波谷很有帮助。要进一步查看，点击图形中的任意点就会看到报文的细节。

　　为了讲解方便，点击示例报文包，或用自己的wireshark点击Statistics – IO Graphs。这个抓包是HTTP下载遇到报文丢失的情况。



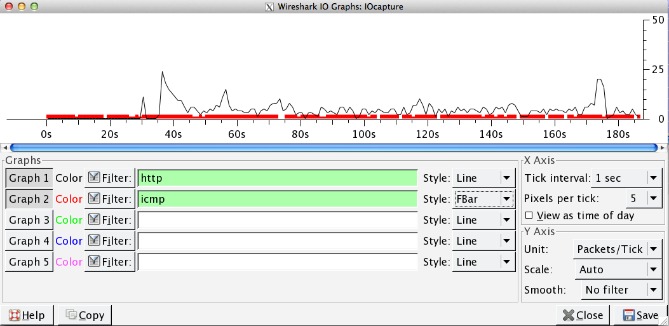
注意：过滤条件为空，此图形显示所有流量。

　　这个默认条件下的显示在大多数troubleshooting中并不是非常有用。将Y轴改为bits/tick这样就可以看到每秒的流量。从这张图可以看到峰值速率是300kbps左右。如果你看到有些地方流量下降为零，那可能是一个出问题的点。这个问题在图上很好发现，但在看报文列表时可能不那么明显。

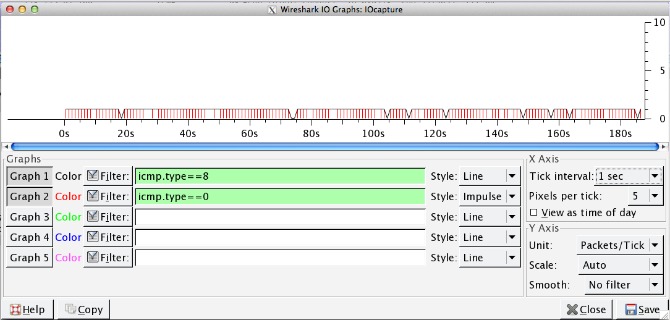


**过滤：**

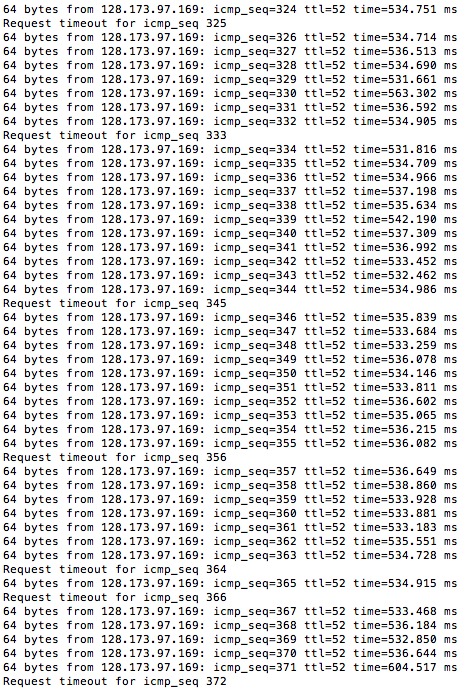
　　每一个图形都可以应用一个过滤条件。这里创建两个不同的graph，一个HTTP一个ICMP。可以看到过滤条件中Graph 1使用“http”Graph 2使用“icmp”。图中可以看到红色ICMP流量中有些间隙，进一步分析。



　　创建两个图形，一个显示ICMP Echo（Type=8）一个显示ICMP Reply（Type=0）。正常情况下对于每一个echo请求会有一个连续的reply。这里的情况是：



可以看到红色脉冲线(icmp type==0 – ICMP Reply)中间有间隙，而整张图中ICMP请求保持连续。这意味着有些reply没有接收到。这是由于报文丢失导致的reply drop。CLI中看到的ping信息如下：



**常用排错过滤条件:**

　　对于排查网络延时/应用问题有一些过滤条件是非常有用的：

**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。

**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。

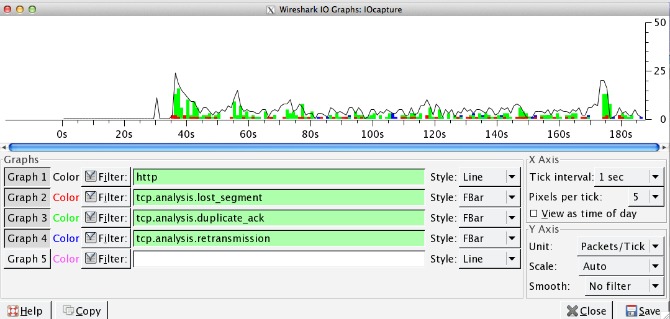
**tcp.analysis.retransmission：**显示抓包中的所有重传。如果重传次数不多的话还是正常的，过多重传可能有问题。这通常意味着应用性能缓慢和/或用户报文丢失。

**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.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。

**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）。

　　在抓包中应用以上一些过滤条件：



　　注意：Graph 1是HTTP总体流量，显示形式为packets/tick，时间间隔1秒。Graph 2是TCP丢失报文片段。Graph 3是TCP 重复ACK。Graph 4是TCP重传。

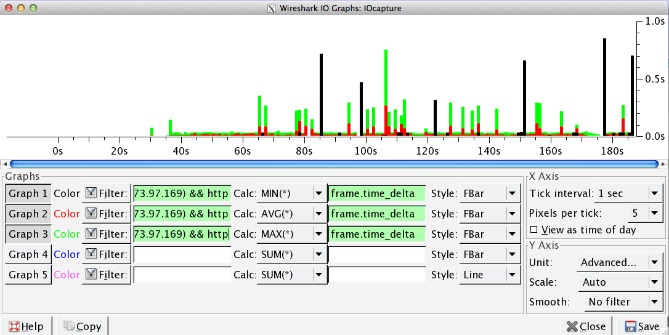
从这张图可以看到：相比于整体HTTP流量，有很多数量的重传以及重复ACK。从这张图中，可以看到这些事件发生的时间点，以及在整体流量中所占的比例。

**函数:**

　　IO Graphs有六个可用函数：SUM, MIN, AVG, MAX, COUNT, LOAD。

**MIN( ), AVG( ), MAX( )**

　　首先看一下帧之间的最小，平均和最大时间，这对于查看帧/报文之间的延时非常有用。我们可以将这些函数结合“**frame.time\_delta”**过滤条件看清楚帧延时，并使得往返延时更为明显。如果抓包文件中包含不同主机之间的多个会话，而只想知道其中一个pair，可将“**frame.time\_delta**”结合源和目标主机条件如“ip.addr==x.x.x.x &&ip.addr==y.y.y.y”。如下图所示：



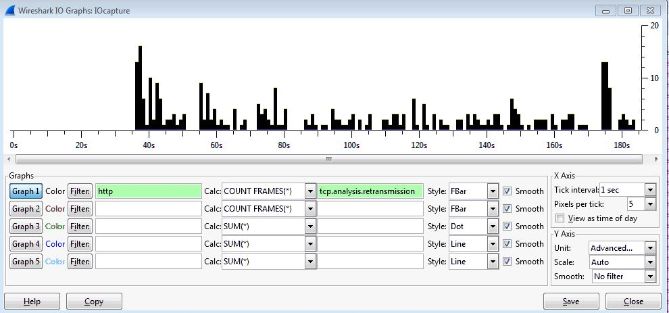
我们做了以下步骤：

* 将Y轴设置为“Advanced”，让Caculation域可见。不做这一步就看不到计算选项。
* X轴时间间隔1秒，所以每个柱状图代表1秒间隔的计算结果。
* 过滤出两个特定IP地址的HTTP会话，使用条件：“(ip.addr==192.168.1.4&& ip.addr==128.173.87.169) && http”。
* 使用3个不同的graph，分别计算Min(), Avg(), Max()。
* 对每一个计算结果应用条件“frame.time\_delta”，将style设置成“FBar”，显示效果最佳。

　　从上图可见，在第106秒时数据流的MAX frame.delta\_time达到0.7秒，这是一个严重延时并且导致了报文丢失。如果想要深入研究，只需要点击图中这一点，就会跳转至相应帧。对应于本例抓包文件中第1003个报文。如果你看见帧之间平均延时相对较低但突然某一点延时很长，可点击这一帧，看看这一时间点究竟发生了什么。

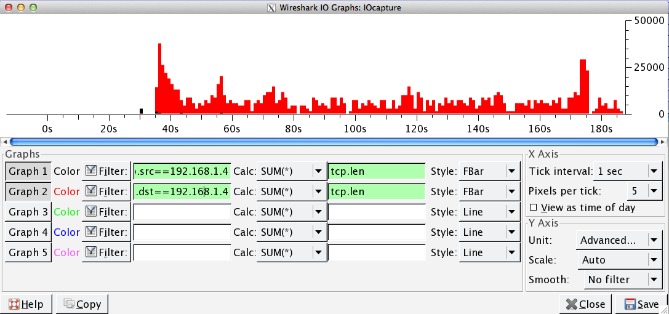
**Count( )**

此函数计算时间间隔内事件发生的次数，在查看TCP分析标识符时很有用，例如重传。例图如下：

[](http://jbcdn2.b0.upaiyun.com/2014/06/6795958e51d21c2ca0cada02ea9f91d2.jpg)

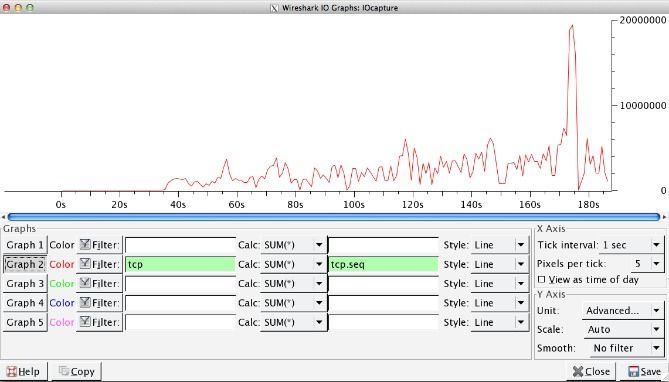
**Sum( )**

　　该函数统计事件的累加值。有两种常见的用例是看在捕获TCP数据量，以及检查TCP序列号。让我们看看第一个TCP长度的例子。创建两个图，一个使用客户端IP 192.168.1.4为源，另一个使用客户端IP作为一个目的地址。每个图我们将sum()功能结合tcp.len过滤条件。拆分成两个不同的图我们就可以看到在一个单一的方向移动的数据量。

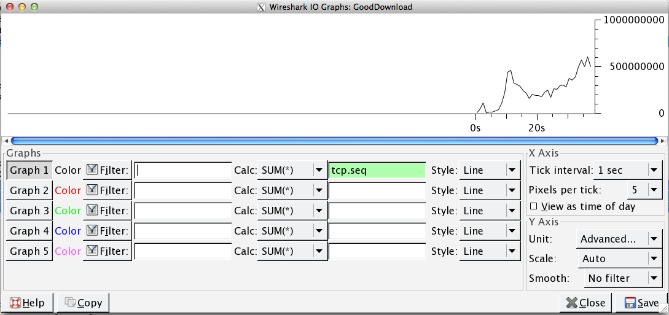


　　从图表中我们可以看到，发送到客户端的数据量（IP.DST = = 192.168.1.4过滤条件）比来自客户端的数据量要高。在图中红色表示。黑条显示从客户端到服务器的数据，相对数据量很小。这是有道理的，因为客户只是请求文件和收到之后发送确认数据，而服务器发送大文件。很重要的一点是，如果你交换了图的顺序，把客户端的IP作为图1的目标地址，并且客户端IP作为图2的源地址，采用了FBAR的时候可能看不到正确的数据显示。因为图编号越低表示在前台显示，可能会覆盖较高图号。

　　现在让我们看一下同一个数据包丢失和延迟的TCP序列号。



　　可以在图中看到若干峰值和下降，表示TCP传输有问题。与正常TCP报文比较：



这张图可以看到TCP序列号相当稳定地增加，表示传输平稳，没有过多重传或丢包。