**Wireshark 抓包工具介绍**

推荐：

<https://www.wireshark.org/>

[https://www.wireshark.org/docs/wsug\_html/#FiltLogOps](https://www.wireshark.org/docs/wsug_html/" \l "FiltLogOps)

TCP-IP详解卷1：协议

1. **Wireshark 是什么**

Wireshark is a network packet analyzer

Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

Wireshark是网络数据包分析器，他是目前免费开源的最好的数据包分析器之一。

**2.有什么功能、特点**

可以在Unix和Windows操作系统使用

可以实时从网络接口抓数据包

可以打开其他抓包工具抓取的数据包文件，进行分析

非常详细的显示数据包，可以显示协议的具体信息

保存或者导出抓取到的数据包

设置很多过滤条件，过滤数据包

根据过滤条件，可以设置颜色显示

对变量进行统计

还有很多

此外Wireshark可以捕获来自许多不同网络媒体类型的流量，包括以太网，无线LAN，蓝牙，USB等详情可以查看[https://wiki.wireshark.org/CaptureSetup/NetworkMedia](https://wiki.wireshark.org/CaptureSetup/NetworkMedia" \t "https://www.wireshark.org/docs/wsug_html_chunked/_top)

**3.注意点**

捕获大文件，需要更多的内存和磁盘空间，因此不用的时候记得关闭

1. **Wireshark的历史**

In late 1997 Gerald Combs started writing Ethereal

In 2006 the project moved house and re-emerged under a new name: Wireshark.

In 2008, Wireshark finally arrived at version 1.0. This release was the first deemed complete, Its release coincided with the first Wireshark Developer and User Conference, called Sharkfest.

1997年末诞生

2006改名为Wireshark

2008 出第一个正式版本

起初是Combs进行开发，目前由个人组织开发和维护

1. **Wireshark的抓包原理**

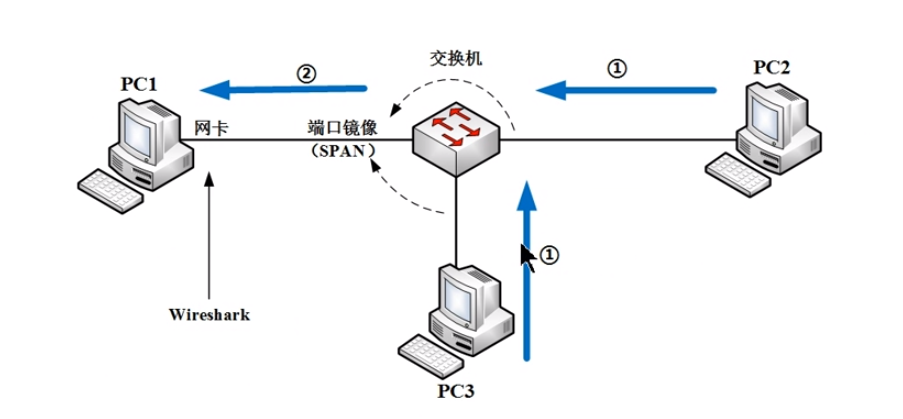
（1）本机环境-直接抓本机网卡进出的流量：直接在终端安装Wireshark，然后Wireshark抓本机网卡的与互联网通信的流量。

libpcap/WinPcap提供了通用的抓包接口，能从不同的类型网络接口获取数据,Wireshark内部有个捕包引擎Capture，利用系统提供的libpcap/WinPcap接口进行抓取数据包。

Wireshark使用libpcap/WinPcap作为接口，直接与网卡进行数据报文交换。

捕获底层的libpcap捕获引擎时，它将从网卡捕获数据包，并将数据包数据保留在（相对）较小的内核缓冲区中。Wireshark读取此数据并将其保存到捕获文件中。

Libpcap相对于Linux平台下，winpcap相对于Windows平台



1. **下载和安装**

<https://www.wireshark.org/download.html>.

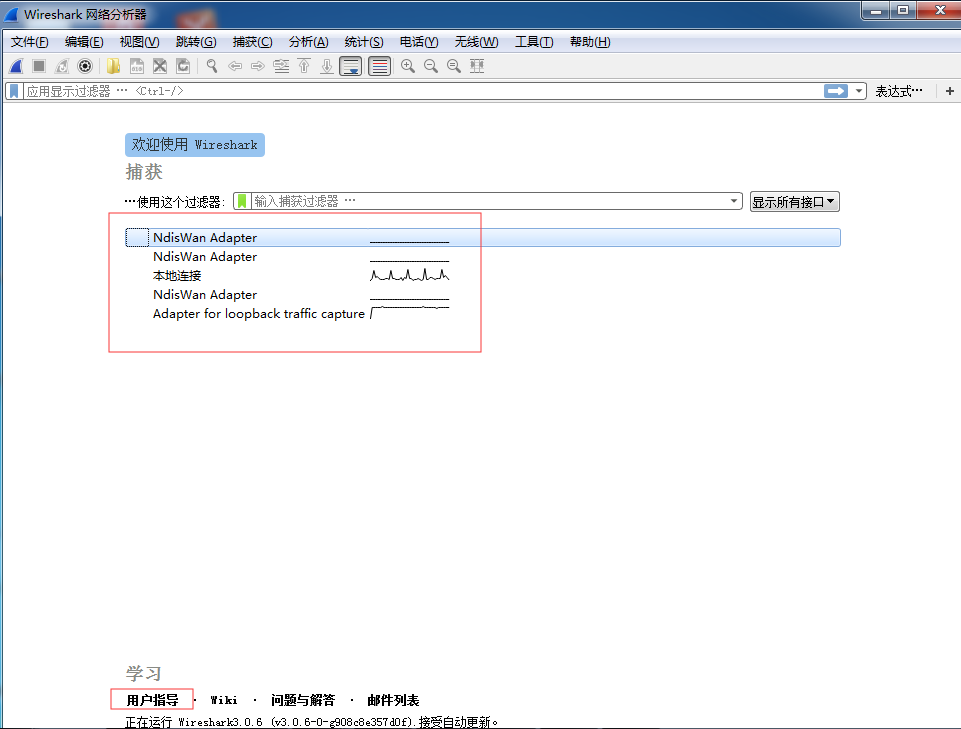
下载后，按照默认的设置进行安装即可。

1. **页面介绍**

**7.1欢迎页面**

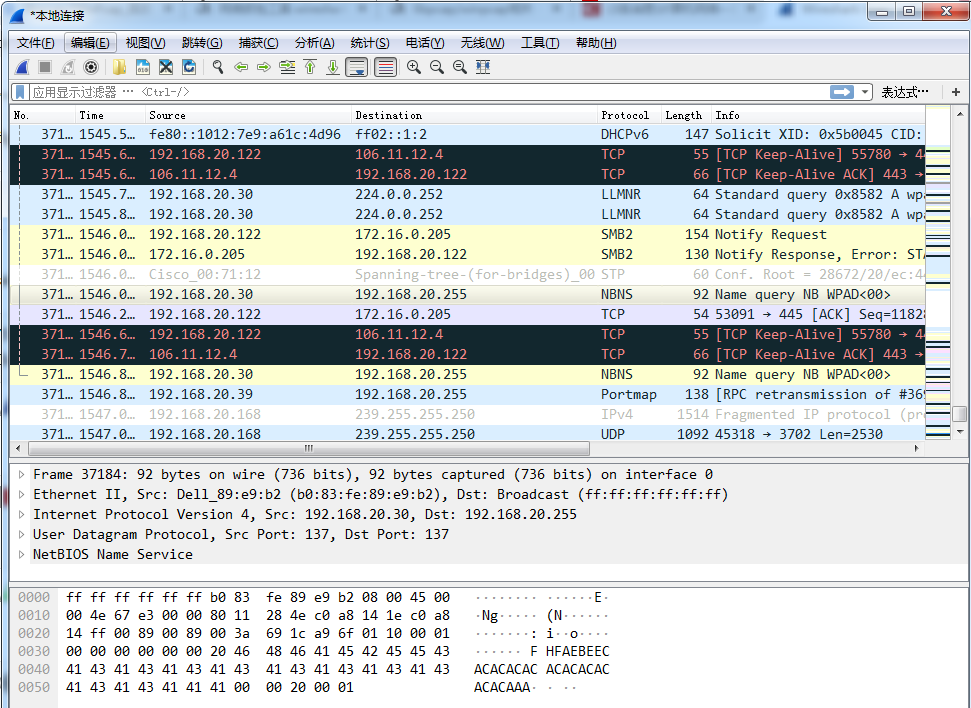
选择网络接口选择：本机连接的是哪个网络，一般右边波浪线的就是本机的连接的网络。

用户指南：点击用户指南，可以进入官网的用户指南网页。



**7.2主页面**

双击网络接口，进入主页面



**7.2.1工具栏**

IMG_256

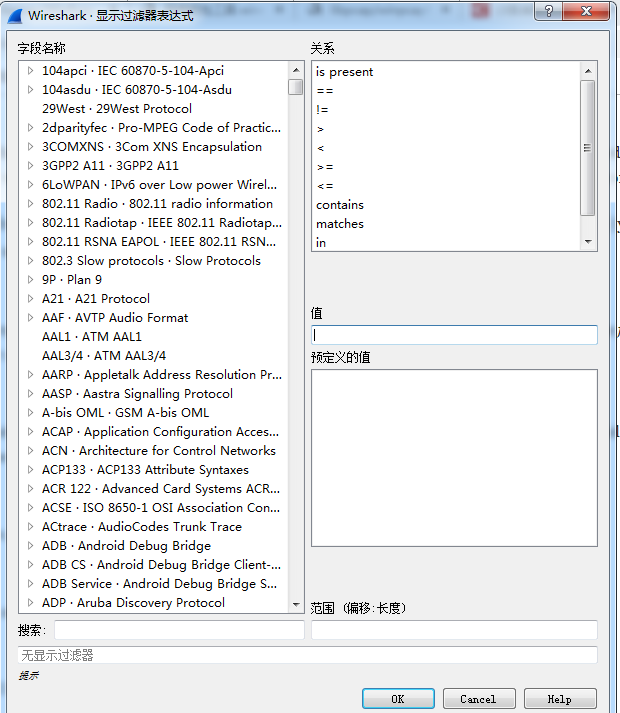
| **Toolbar Icon** | **Toolbar Item** | **Menu Item** | **Description** |
| --- | --- | --- | --- |
| IMG_256 | Start | Capture → Start | 开始捕获数据 |
| IMG_257 | Stop | Capture → Stop | 停止数据捕获 |
| IMG_258 | Restart | Capture → Restart | 重新捕获当前数据 |
| IMG_259 | Options…​ | Capture → Options…​ | 打开捕获数据的设置选项窗口 |
| IMG_260 | Open…​ | File → Open…​ | 打开文件捕获的文件或者tcpdump文件，进行分析 |
| IMG_261 | Save As…​ | File → Save As…​ | 保存当前文件 |
| IMG_262 | Close | File → Close | 关闭捕获文件 |
| IMG_263 | Reload | View → Reload | 重新加载当前捕获文件 |
| IMG_264 | Find Packet…​ | Edit → Find Packet…​ | 查找 |
| IMG_265 | Go Back | Go → Go Back | 向后，向上移动一行 |
| IMG_266 | Go Forward | Go → Go Forward | 向前，向前移动一行 |
| IMG_267 | Go to Packet…​ | Go → Go to Packet…​ | 跳转到特殊的数据包 |
| IMG_268 | Go To First Packet | Go → First Packet | 跳转到第一个抓到的数据包 |
| IMG_269 | Go To Last Packet | Go → Last Packet | 跳转到捕获的最后一个数据包 |
| IMG_270 | Auto Scroll in Live Capture | View → Auto Scroll in Live Capture | 自动滑动到最新的数据 |
| IMG_271 | Colorize | View → Colorize | 是否根据颜色来分数据 |
| IMG_272 | Zoom In | View → Zoom In | 放大 |
| IMG_273 | Zoom Out | View → Zoom Out | 缩小 |
| IMG_274 | Normal Size | View → Normal Size | 正常大小 |
| IMG_275 | Resize Columns | View → Resize Columns | 调整表头的行间距 |

**7.2.2过滤工具栏**

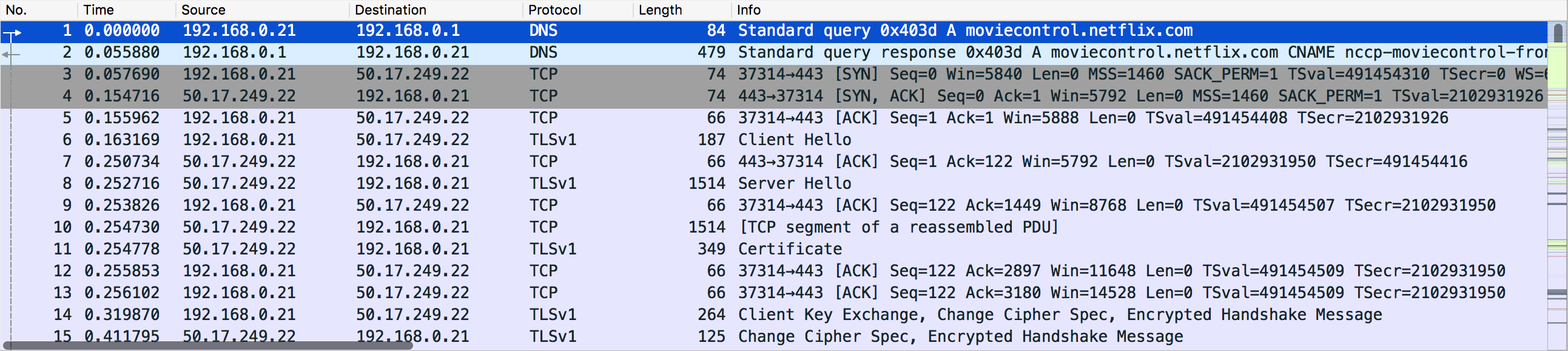
IMG_256

| **Toolbar Icon** | **Name** | **Description** |
| --- | --- | --- |
| IMG_256 | Bookmarks | 管理或者保存过滤条件 |
| IMG_257 | Filter Input | 输入过滤条件 |
| IMG_258 | Clear | 清除过滤条件 |
| IMG_259 | Apply | 启用当前的过滤条件 |
| IMG_260 | Recent | 选择最近的过滤列表 |
| Expression…​ | Filter Expression | 打开过滤表示试的对话框 |
| IMG_261 | Add Button | 添加一个新的过滤表达式按钮 |
| Squirrels | Expression Button | 过滤表达式例子 |

关系表达式图片如下：



**7.2.3“数据包列表”窗格**



No：序号，按照0开始

Time： 时间戳，可以根据根式进行修改

Source：源IP,数据从哪发出

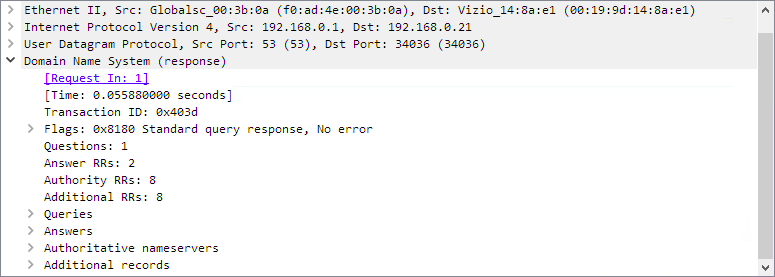
Destination：目标IP，数据要发到哪

Protocol： 显示协议名和协议版本简写

Length： 数据包长度

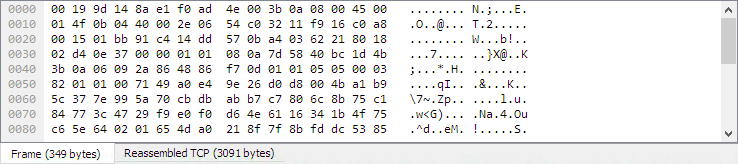
Info：数据包信息详情

**7.2.4“数据包详细信息”窗格**



此窗格显示在“数据包列表”窗格中选择的数据包的协议和协议字段

**7.2.5带标签的“数据包字节”窗格**



数据包左边按照十六进制显示，右边按照ascii显示， 这个数据包并不只是我们发送的数据，它按照不同的网络层的协议对数据包进行组包

1. **过滤数据包**

如何过滤数据，主要是根据以下内容显示数据包

Protocol：协议

The presence of a field：协议对应的字段

The values of fields：字段的值

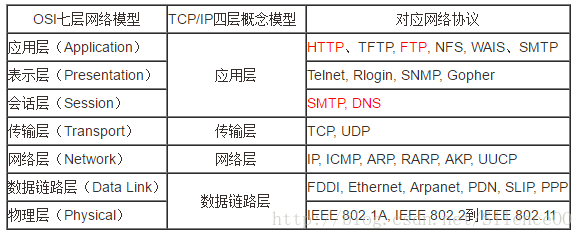
A comparison between fields：字段之间的关系

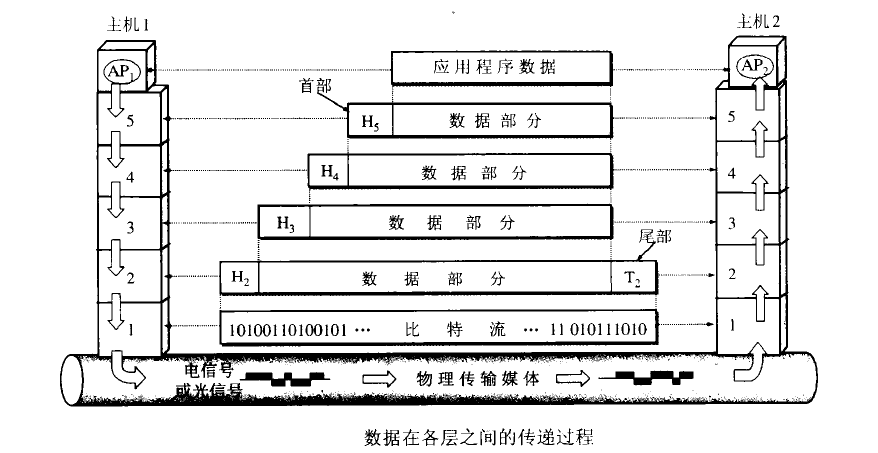
…​ and a lot more!

我们先考虑过滤的对象是什么，他有什么内容？ 过滤的对象就是数据包，数据包是怎么组成的呢？

8.1 数据包结构

(1)网络层次结构

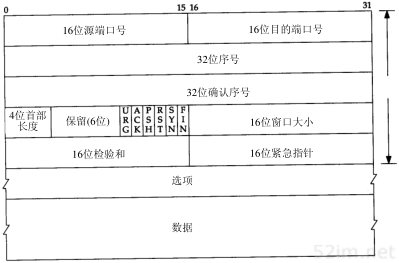




从上图可知，Socket 是基于TCP的实现，发出去的数据需要经过传输层、网络层和数据链路层的数据重组。

（2）TCP协议部首

下图显示TCP首部的数据格式。如果不计任选字段，它通常是20个字节。从下图可以看出TCP首部数据主要包括端口、序号、确认序号、部首长度、标志位、窗口大小、校验和、紧急指针。从这些数据来看，可以从端口、标志位过滤，因为其他都是变化的不适合做过滤条件。



顺便对标志进行说明：

SYN: 同步序号用来发起一个连接，这个标志

URG：紧急指针有效

ACK：确认序号有效

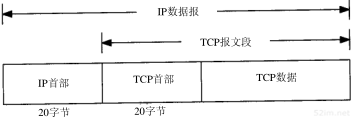
PSH：接收方应该尽快将这个报文段交给应用层

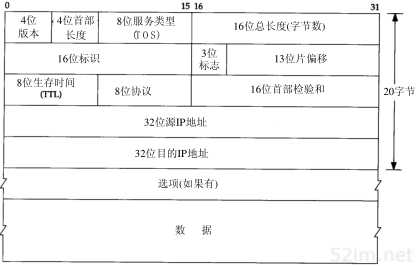
RST: 重新建立连接

FIN: 发端完成发送任务

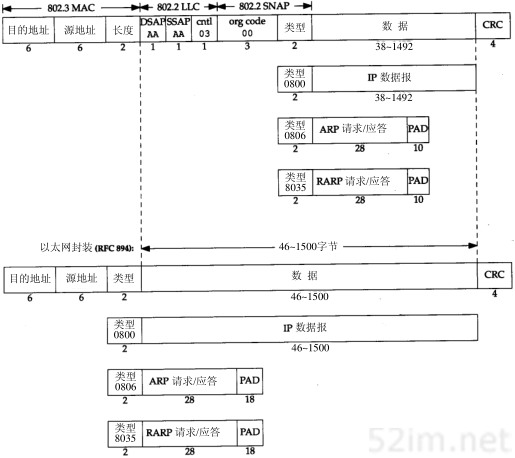
（3）IP协议部首

下图显示IP部首，我们可以采用源IP地址进行过滤





（4）下图显示链路层部首和部尾 ,我们可以采用源MAC地址进行过滤



**8.2 过滤操作符**

wireshark的过滤项可以是“协议“+”.“+”协议字段”的模式。以端口为例，端口出现于tcp协议中所以有端口这个过滤项且其写法就是tcp.port ，实际使用时我们输入“协议”+“.”wireshark就会有支持的字段提示。

**English**和**C-like**运算符是等价的，并且可以在过滤字符串中混合使用。

| **English** | **C-like** | **Description** | **Example** |
| --- | --- | --- | --- |
| eq | == | Equal | ip.src==10.0.0.5 |
| ne | != | Not equal | ip.src!=10.0.0.5 |
| gt | > | Greater than | frame.len > 10 |
| lt | < | Less than | frame.len < 128 |
| ge | >= | Greater than or equal to | frame.len ge 0x100 |
| le | <= | Less than or equal to | frame.len <= 0x20 |
| contains |  | Protocol, field or slice contains a value | sip.To contains "a1762" |
| matches | ~ | Protocol or text field matches a Perl-compatible regular expression | http.host matches "acme\.(org|com|net)" |
| bitwise\_and | & | Bitwise AND is non-zero | tcp.flags & 0x02 |

混合表达式

| **English** | **C-like** | **Description** | **Example** |
| --- | --- | --- | --- |
| and | && | Logical AND | ip.src==10.0.0.5 and tcp.flags.fin |
| or | || | Logical OR | ip.scr==10.0.0.5 or ip.src==192.1.1.1 |
| xor | ^^ | Logical XOR | tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29 |
| not | ! | Logical NOT | not llc |
| […​] |  | Subsequence | See “Slice Operator” below. |
| in |  | Set Membership | http.request.method in {"HEAD" "GET"}. See “Membership Operator” below. |

**8.2 常用过滤条件**

（1）过滤端口

tcp.port == 8000

tcp.port eq 8000

tcp.port in [8000 9000]

（2）过滤TCP标志

tcp.flags.syn == 1

tcp.flags.fin == 1

tcp.flags.urg == 1

tcp.flags.reset == 1

（2）过滤ip

ip.addr == 192.168.1.1

ip.src == 10.0.0.5

ip.des == 10.0.0.5

1. 网卡地址（MAC地址）

eth.addr == 78:e3:b5:bc:a9:04

eth.dst == 78:e3:b5:bc:a9:04

eth.src == 78:e3:b5:bc:a9:04

1. 组合过滤

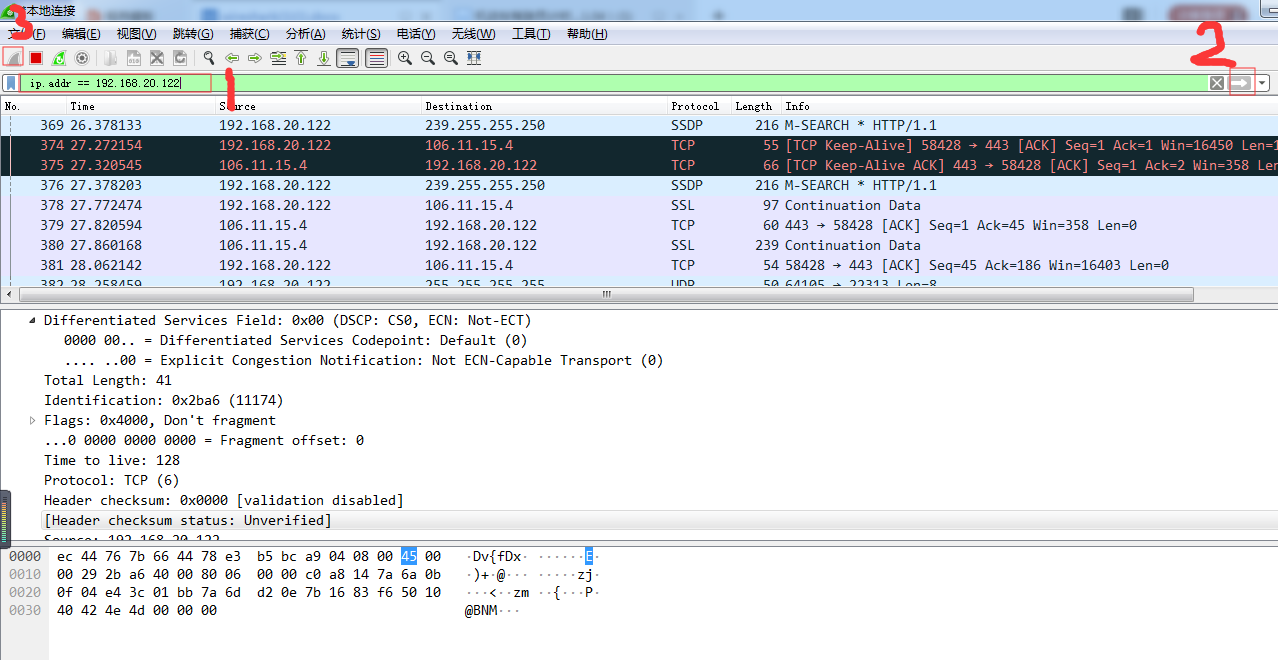
tcp.port == 443 and ip.addr == 192.168.20.122

tcp.port == 443 and eth.addr == 78:e3:b5:bc:a9:04

**8.3 过滤例子**

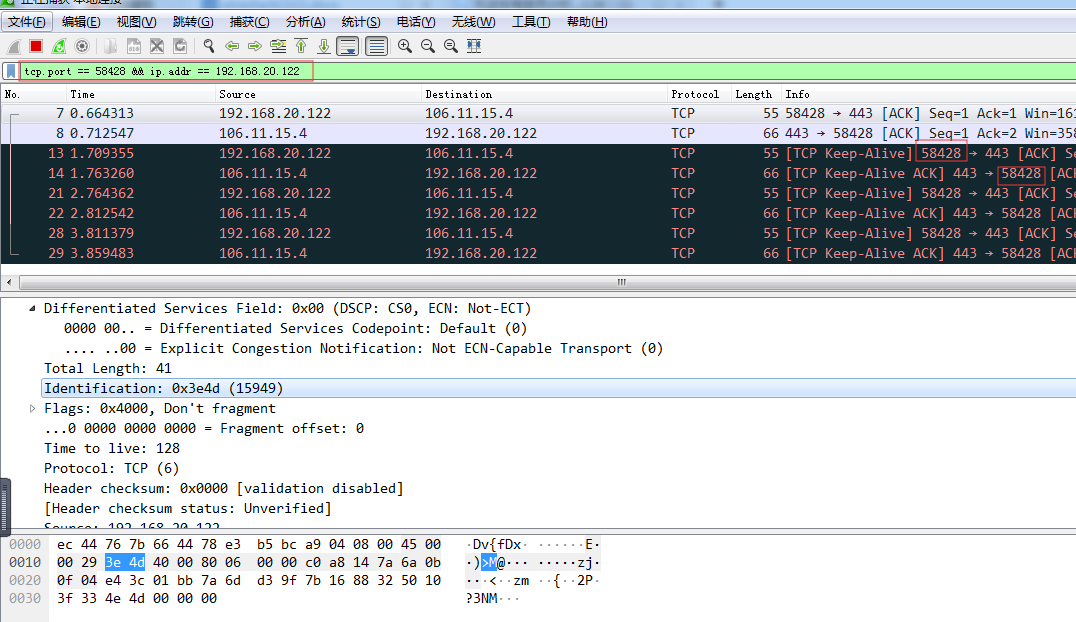
**8.3.1 过滤ip**

在过滤输入框输入顾虑条件 ip.addr == 192.168.20.122 ，点击右边的箭头应用此过滤条件，点击3图标开始捕获数据



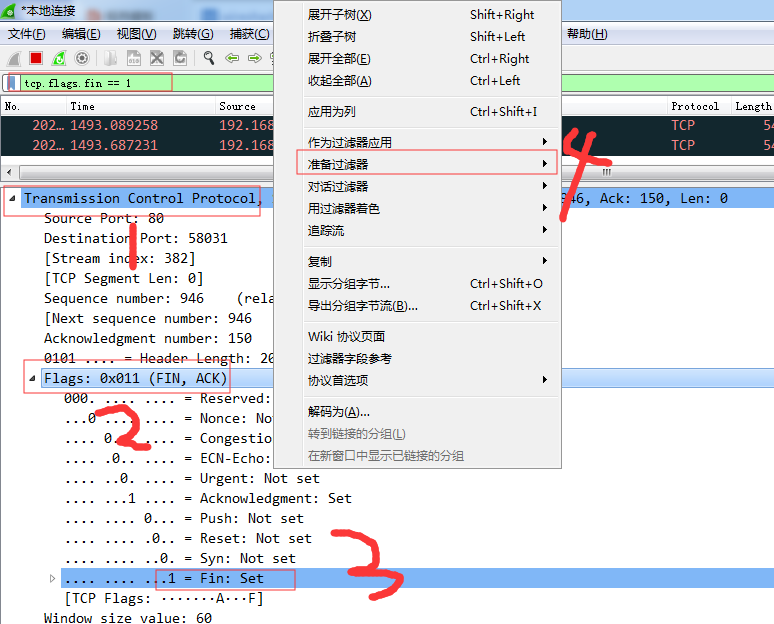
**8.3.2 过滤端口和IP**

tcp.port == 58428 && ip.addr == 192.168.20.122



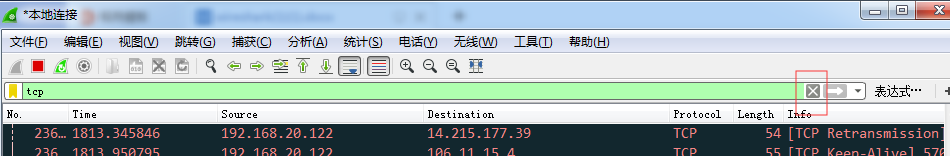
**8.3.3 通过协议窗口过滤**

还有快捷的过滤方式，在协议窗口 ，打开你要过来的协议，选择要过滤的属性->右键弹出对话框，点击准备过滤器->选中 ->在过滤输入栏就会显示要过滤的内容,点击启用过滤按钮即可。



**8.3.4 清除过滤条件**

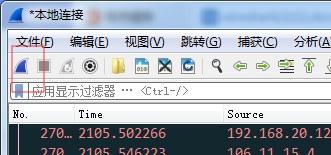
点击过滤工具栏的“X”按钮即可



**8.4抓包**

**8.4.1 启动抓包**

点击下方的图标 或者捕获菜单->开始



**8.4.2 暂停抓包**

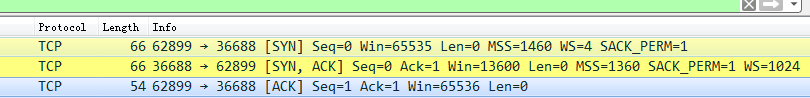
点击下方的图标 或者捕获菜单->停止

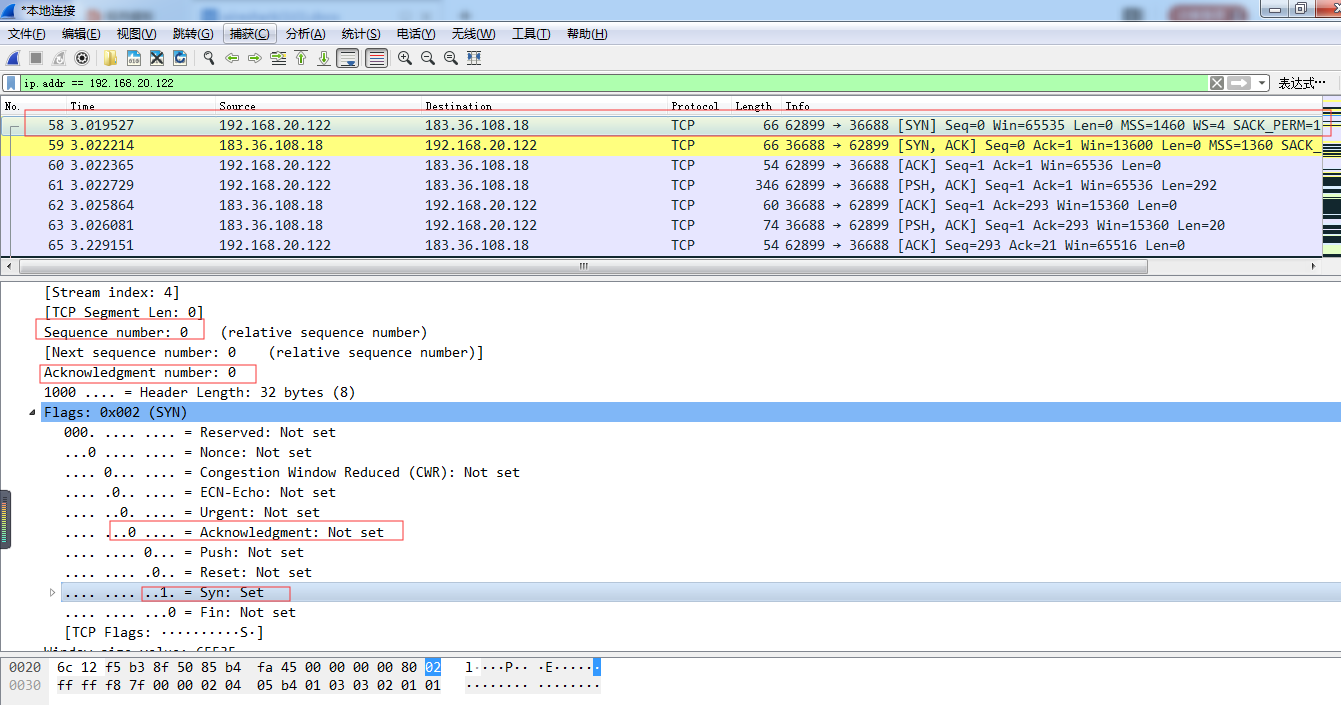


**8.5 分析捕获数据**

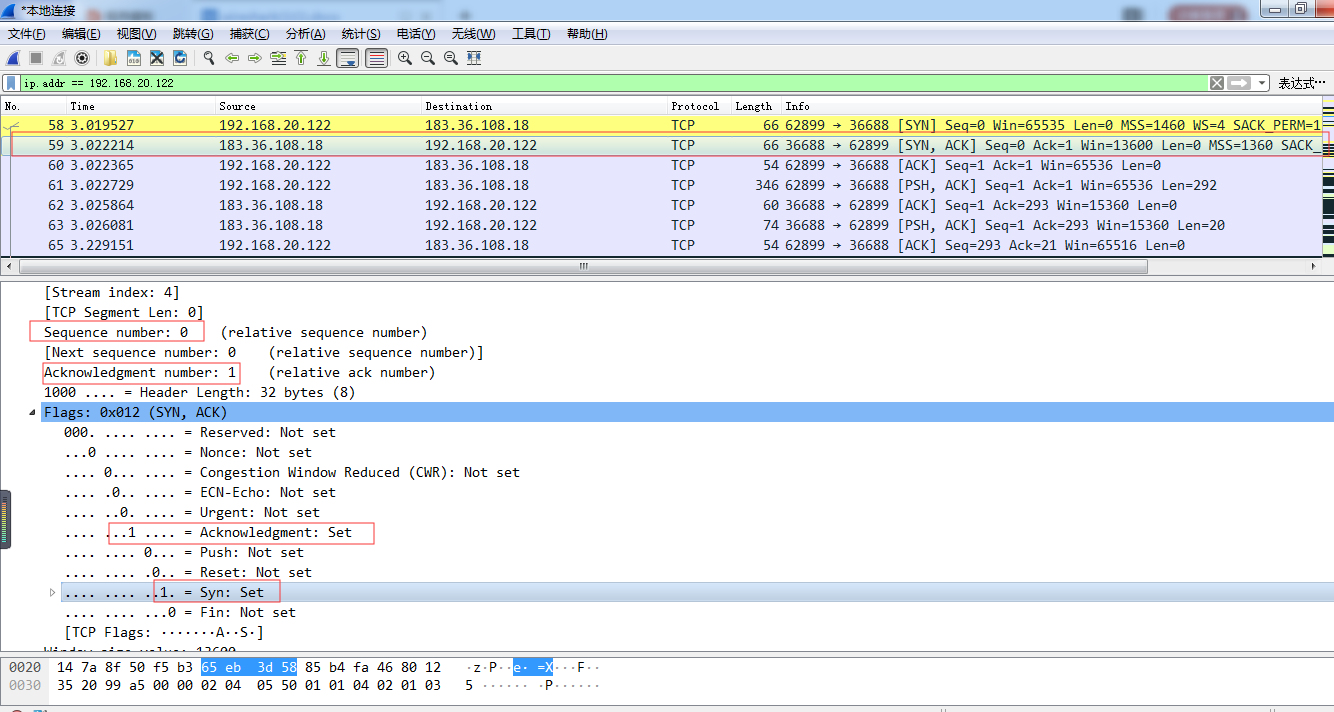
**8.5.1 创建连接三次握手**

通过标识SYN来查看连接创建

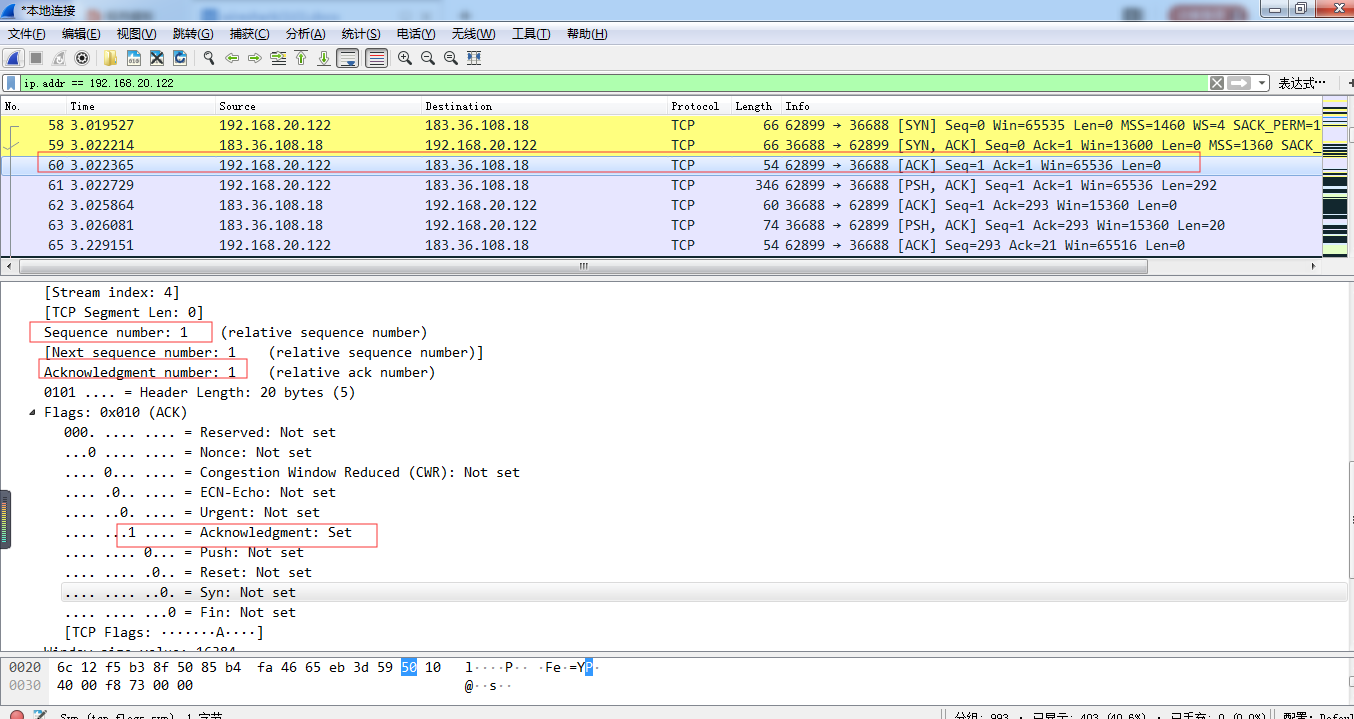




（1）请求端（通常称为客户）发送一个SYN段指明客户打算连接的服务器的端口，以及初始序号seq=0,SYN=1。

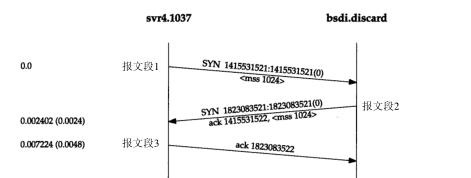


（2）服务器发回包含服务器的初始序号的SYN报文段（报文段2）作为应答。同时，将确认序号设置为客户的ISN加1以对客户的SYN报文段进行确认。一个SYN将占用一个序号。



（3）客户必须将确认序号设置为服务器的ISN加1以对服务器的SYN报文段进行确认（报文段3）。

这三个报文段完成连接的建立。这个过程也称为三次握手（three-way handshake）。



**8.5.2 关闭连接，四次挥手**

通过标识FIN来查看连接关闭



报文段4发起终止连接，它由Te lnet客户端关闭连接时发出。这在我们键入quit命令后发生。它将导致TCP客户端发送一个FIN，用来关闭从客户到服务器的数据传送。

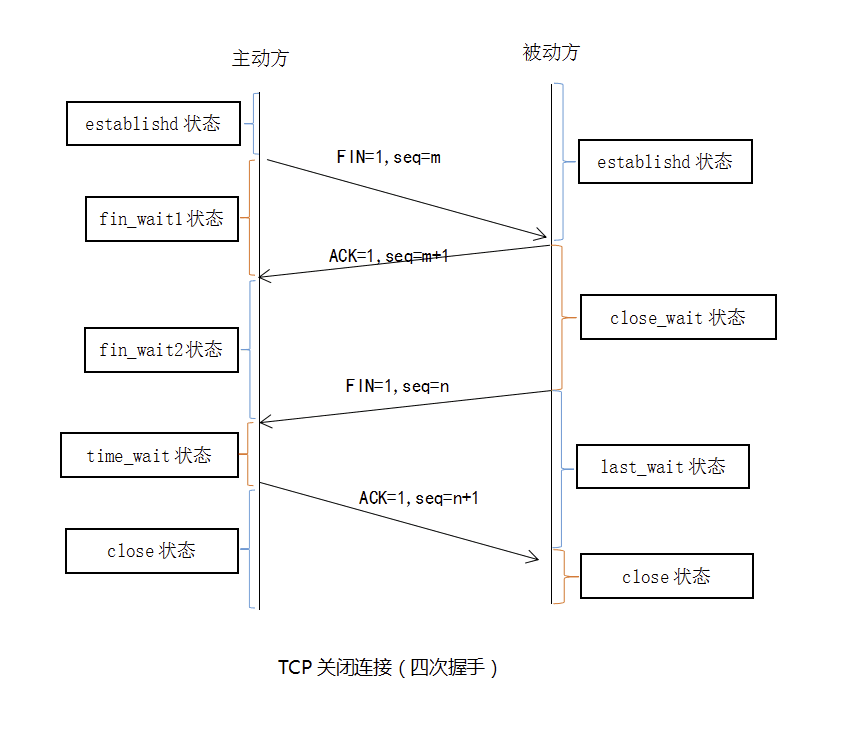
当服务器收到这个FIN，它发回一个ACK，确认序号为收到的序号加1（报文段5）。和SYN一样，一个FIN将占用一个序号。同时TCP服务器还向应用程序（即丢弃服务器）传送一个文件结束符。接着这个服务器程序就关闭它的连接，导致它的TCP端发送一个FIN（报文段6），客户必须发回一个确认，并将确认序号设置为收到序号加1（报文段7）。

（1）客户端发送关闭连接

（2）平台收到关闭连接请求，并回复ACK确认收到请求

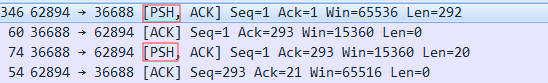
（3）服务器发送FIN关闭连接

（4）客户端收到服务器的FIN后，回复ACK确认收到请求



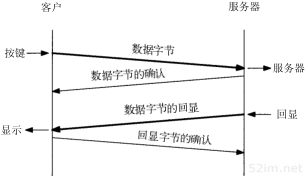
**8.5.3 发送数据**

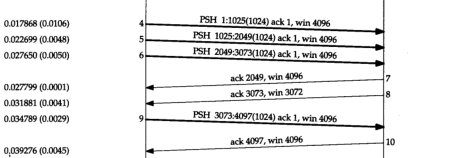
发送数据用PSH标志 ，确认回复有2种情况，一种是收到就回复，另外一种是收到后等缓冲满了再回复。



（1）客户端发送PSH发数据，数据长度是Len ,Seq=1

（2）服务器接收到PSH后，回复确认，ACK=客户端的Seq+长度

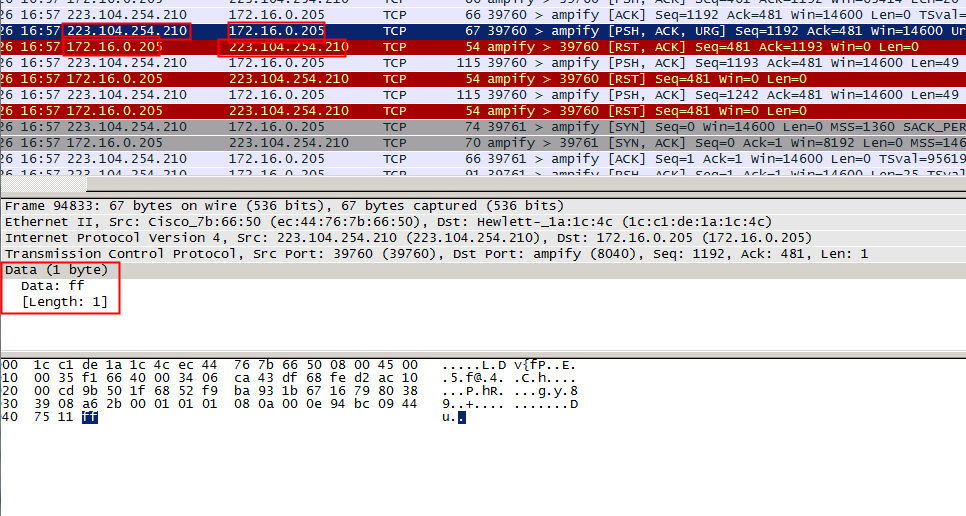


使用TCP的滑动窗口协议时，接收方不必确认每一个收到的分组

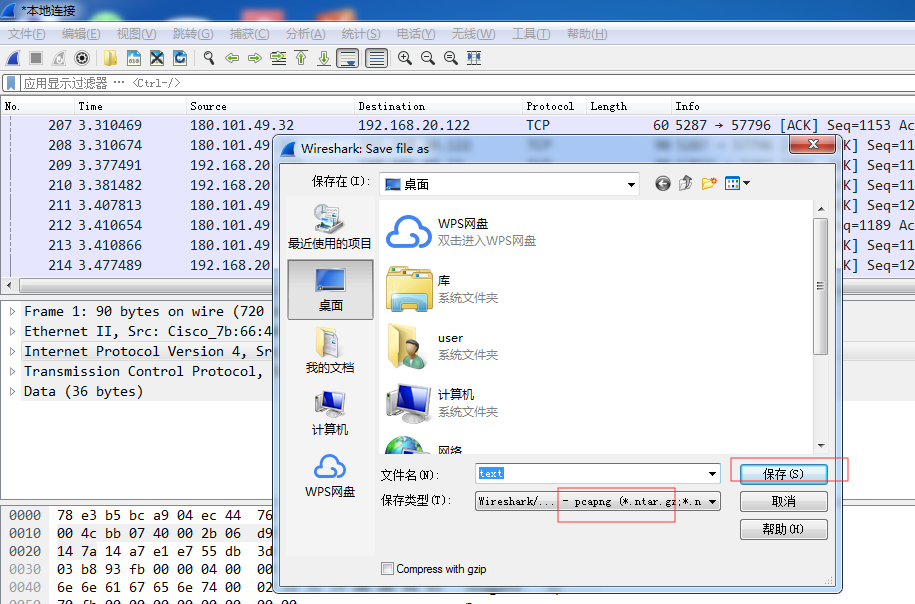
**8.5.4 连接异常，网络重置**

网络重置RST，当网络出现异常，会出现网络重置的情况

下图出现网络重置的原因是发了一个紧急包URG

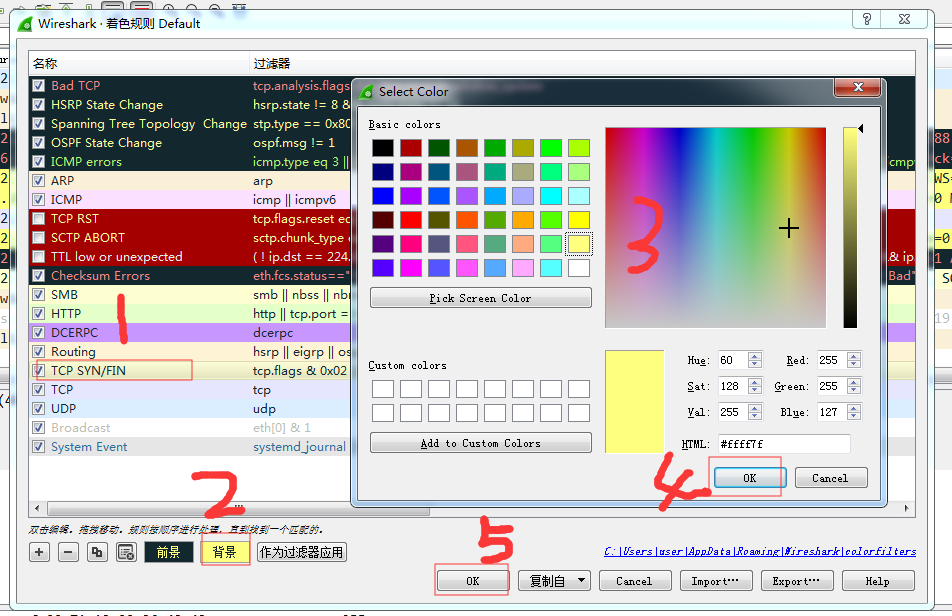


**9. 保存数据**

文件 → 保存或文件 → 另存为...菜单项。您可以选择要保存的数据包以及要使用的文件格式。

**10.数据包颜色设置**

视图->着色规则->左键点击要修改颜色的行->背景->选择一种背景颜色->OK->OK



**11.分析图标**

**11.1往返时间图**

往返时间（round-trip time）它是可以用来显示一个数据包从发数据到收到确认收到一个过程话费的时间，用来观察在哪些时间网络稳定情况，还可以压测看看服务器的瓶颈。

统计-TCP流图像-往返时间

