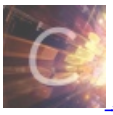
山东大学计算机网络实验——Protocol Layers

原创

PancrasBohemian 于 2017-11-04 20:10:25 发布 6843 收藏 67

分类专栏： 助教笔记 文章标签： wireshark

本文链接：https://blog.csdn.net/PancrasBohemian/article/details/78445635



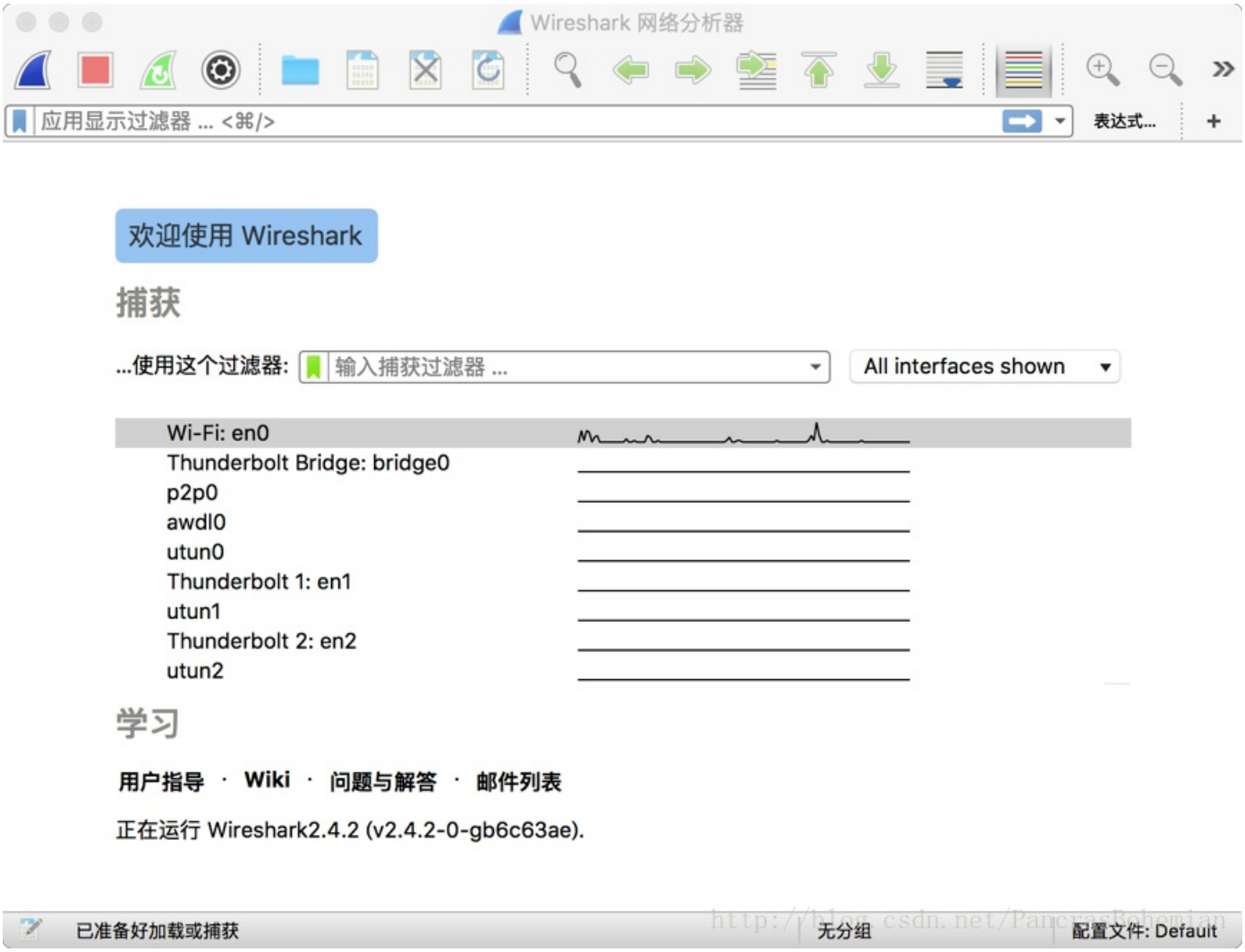
山东大学计算机网络实验一 Protocol Layers

发现实验手册上的WireShark版本还停留在1.8、1.9，所以写一篇在Mac上进行实验的教程，希望能够帮到其他同学

Step1

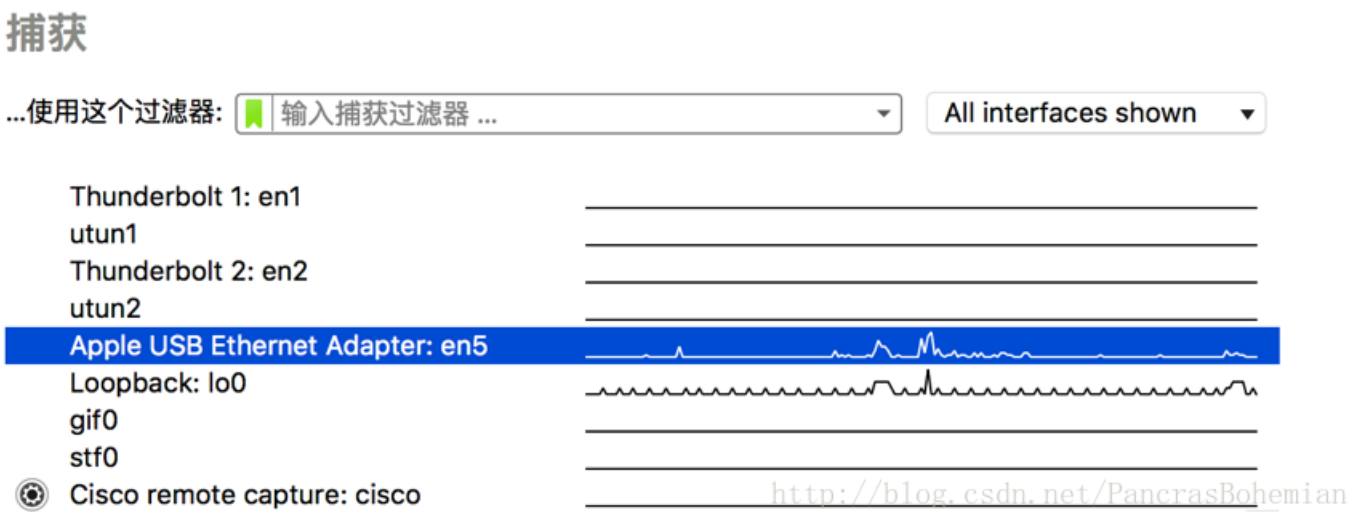
使用WireShark 抓包分析。

首先打开WireShark，看到这个界面



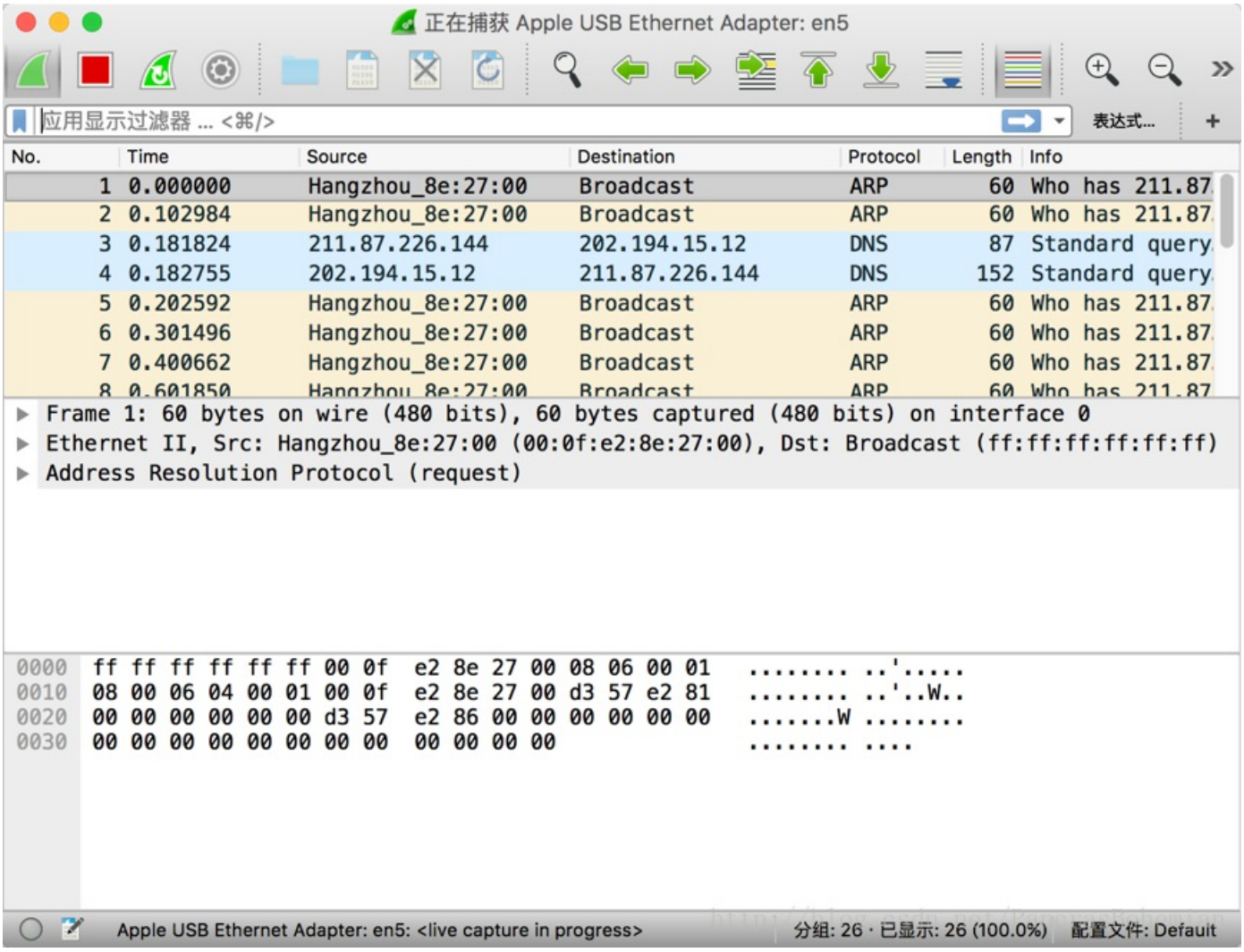
你要确定你自己使用的网络是哪一个，因为Mac都没有网线插口，所以我们用网线的话都需要一个Adapter（适配器）。
我们打开系统偏好设置->网络

我用的适配器是Apple官方的，所以名字叫Apple USB Ethernet Adapter，我们在WireShark中选择这个，你们在用的时候选择正确的即可。

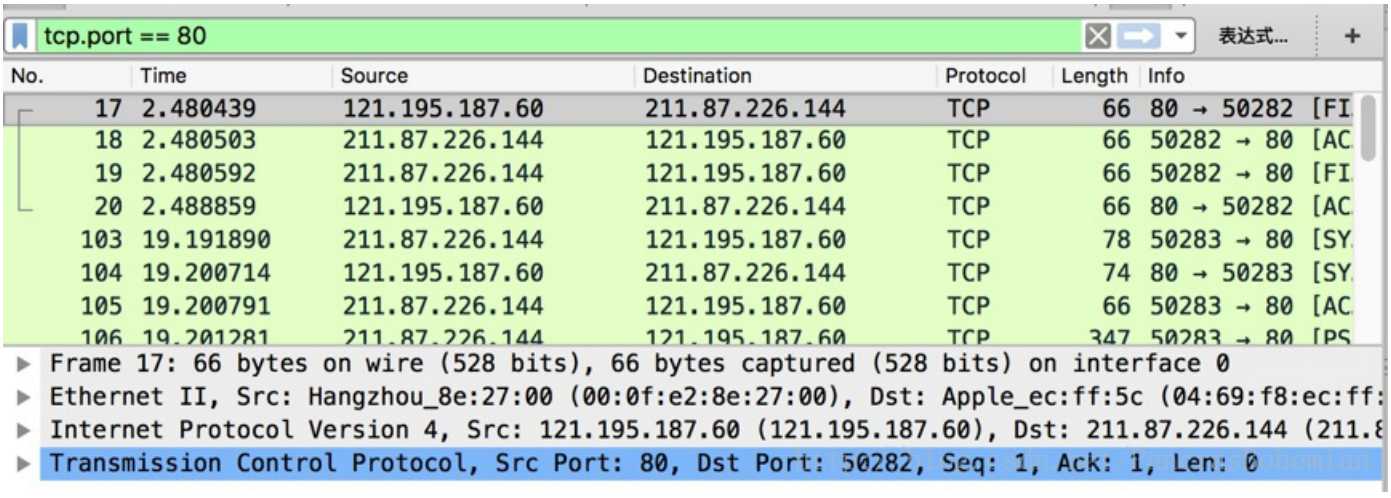


选择后出现了这个页面

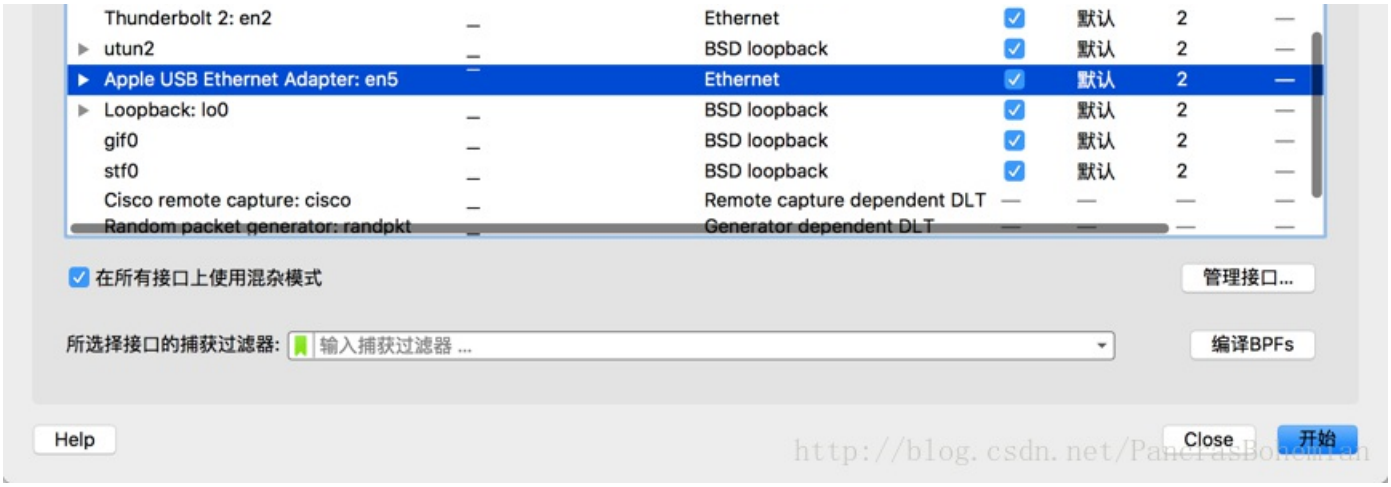
选择后出现下这个页面



随后我们在上方的输入框中输入tcp.port == 80 不是手册上写的 tcp port 80！！！

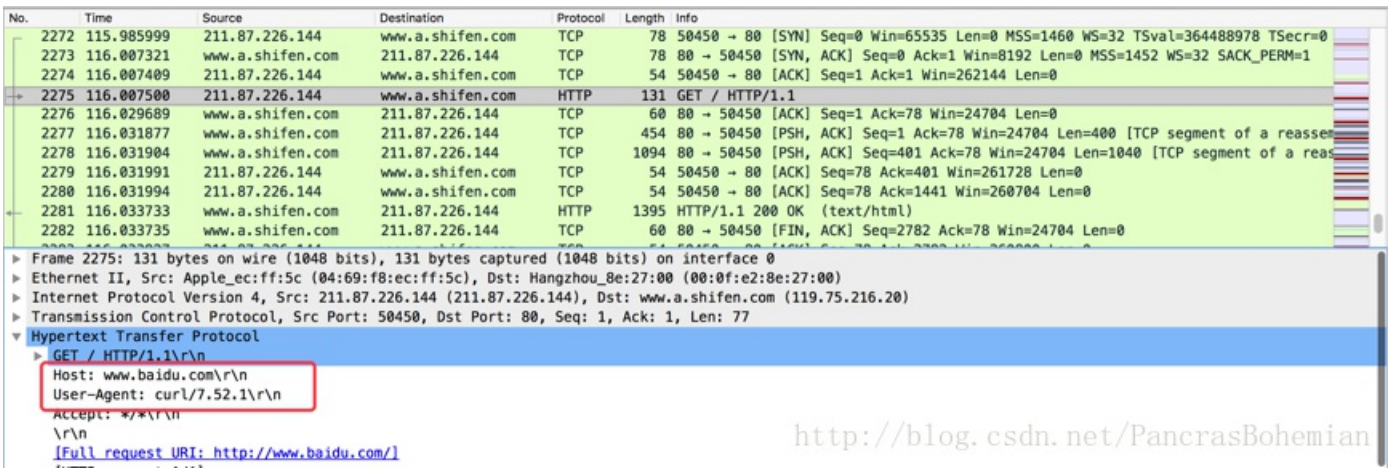


看到实验手册上要求check “enable network name resolution”.
其实没那么玄乎，首先停止获取，然后使用cmd+K或者在上方菜单栏中选择捕获->选项



在这里确保你选择了正确的接口之后，点上方的选项

发现已经选好了……费了好多劲找到，那我们那就点击start开始吧。
开始之后我们继续按照手册上来，在命令行中输入curl http://www.baidu.com
然后我们发现了海量的包，哪些是我们需要的呢？就是那些绿色底的包



Host是百度
UserAgent是我们使用的curl，这就没错啦，完成了第一步。

Step 2

这一步主要是根据上一步来，通过对不同包（这个定义不准确，一组信息在链路层被称作帧，在网络层被称作包，在传输层称作段，在应用层称作消息）的拆分等操作，了解里面的信息。
首先找到一个协议为HTTP的包

No.	Time	Source	Destination	Protocol	Length	Info
2272	115.985999	211.87.226.144	www.a.shifen.com	TCP	78	50450 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2273	116.007321	www.a.shifen.com	211.87.226.144	TCP	78	80 → 50450 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=
2274	116.007409	211.87.226.144	www.a.shifen.com	TCP	54	50450 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
2275	116.007500	211.87.226.144	www.a.shifen.com	HTTP	131	GET / HTTP/1.1
2276	116.029689	www.a.shifen.com	211.87.226.144	TCP	60	80 → 50450 [ACK] Seq=1 Ack=78 Win=24704 Len=0
2277	116.031877	www.a.shifen.com	211.87.226.144	TCP	454	80 → 50450 [PSH, ACK] Seq=1 Ack=78 Win=24704 Le
2278	116.031904	www.a.shifen.com	211.87.226.144	TCP	1094	80 → 50450 [PSH, ACK] Seq=401 Ack=78 Win=24704
2279	116.031991	211.87.226.144	www.a.shifen.com	TCP	54	50450 → 80 [ACK] Seq=78 Ack=401 Win=261728 Len=
2280	116.031994	211.87.226.144	www.a.shifen.com	TCP	54	50450 → 80 [ACK] Seq=78 Ack=1441 Win=260704 Len
2281	116.033733	www.a.shifen.com	211.87.226.144	HTTP	1395	HTTP/1.1 200 OK (text/html)

这个包有着GET方法，就是我们要第一步找到的包，也就是No.2275包，这个包是从我们的电脑发送到服务器上的。

```
▶ Frame 2275: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits) on interface 0
▶ Ethernet II, Src: Apple_ec:ff:5c (04:69:f8:ec:ff:5c), Dst: Hangzhou_8e:27:00 (00:0f:e2:8e:27:00)
▶ Internet Protocol Version 4, Src: 211.87.226.144 (211.87.226.144), Dst: www.a.shifen.com (119.75.216.20
▶ Transmission Control Protocol, Src Port: 50450, Dst Port: 80, Seq: 1, Ack: 1, Len: 77
▶ Hypertext Transfer Protocol
```

我们从应用中间的那一层可以看到这个包的一些细节。
第一部分是Frame，这不是一个协议，只是一个包的概括性信息的集合。
第二部分是Ethernet，这个就开始和下图匹配了

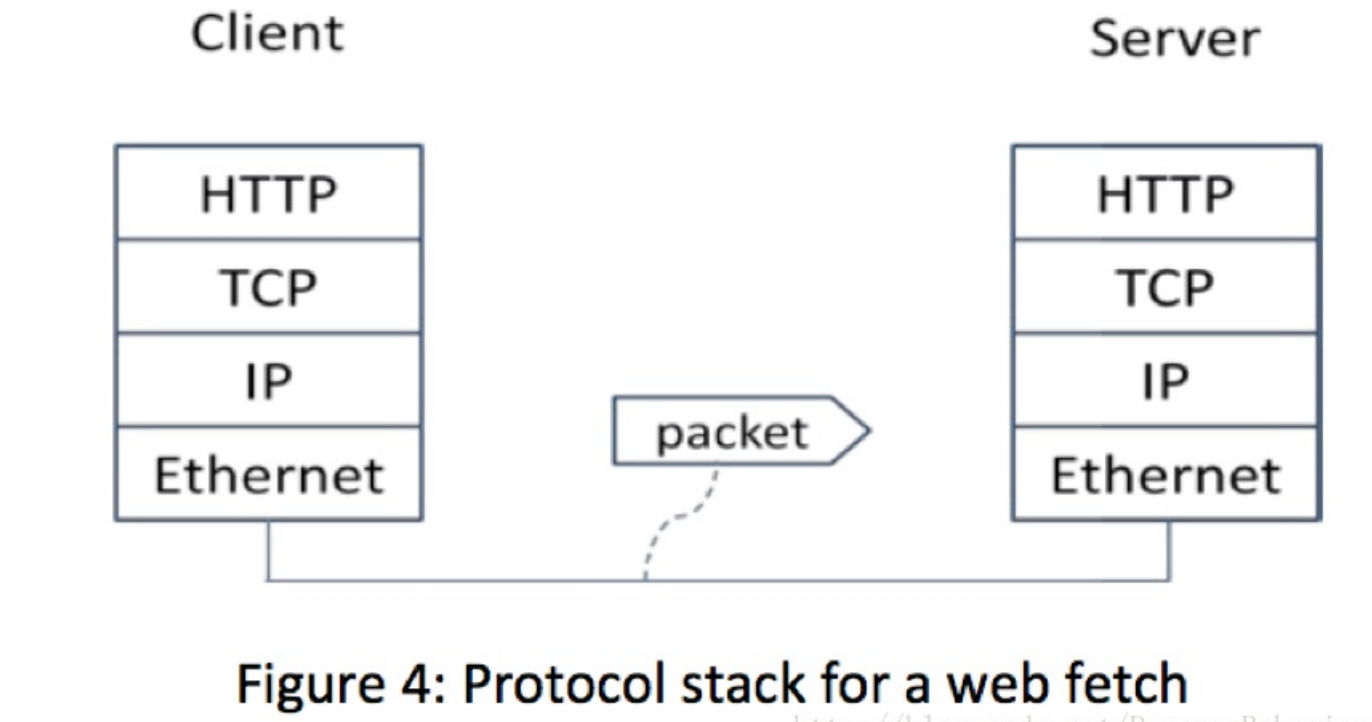
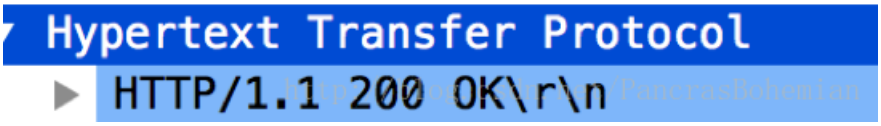


Figure 4: Protocol stack for a web fetch

随后就是经典的IP、TCP、HTTP三个协议，我们注意到，这个顺序是从栈底到栈顶依次上升的，这是因为包是从顶到底依次传递的，低层次协议的头部信息被加到高层次协议信息的前头。具体可以见课本（中文版P25）。
这一步结束之后，我们找到另一个HTTP包
也就是2281号包，我们注意到在No这一行，两个HTTP包都带着一个箭头，指明方向，我们可以明白，这个包就是从服务器到我们的电脑的。与上一个HTTP包相比，这个包里面包含着“200 OK”的信息

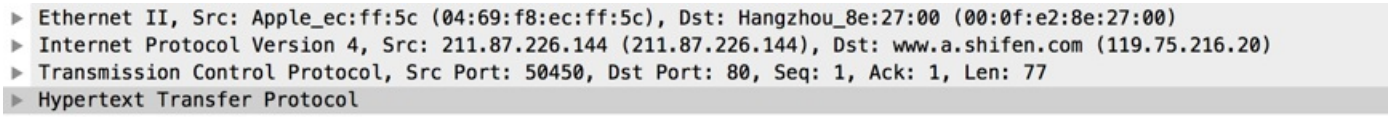


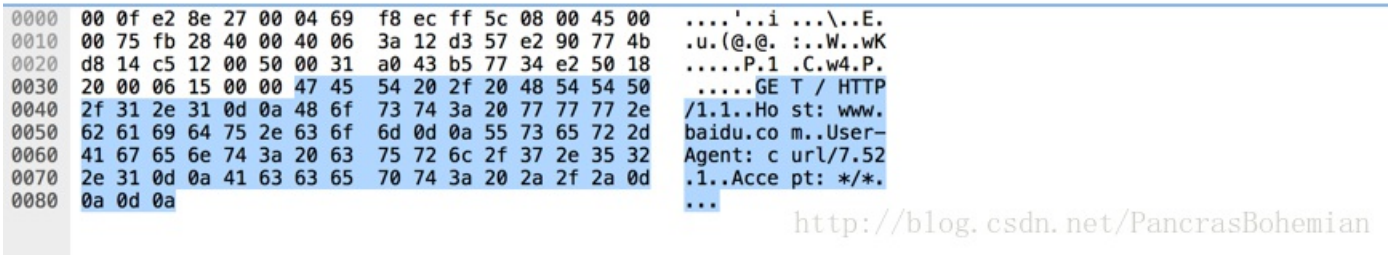
这个信息说明我们的获取是成功的。
我们同时发现，这个面板中多出来了两个块，也就是下图中没有标蓝的两块

上面那一块描述了不仅仅包含这个包的信息。在最可能的情况下，网络请求被以一系列包的方式通过网络传输，并最终在电脑上被组装成信息。标有HTTP的包是网络请求中的最后一个包，并且能列出可以获得完整网络请求的所有的包。
下面那一块描述了网页获取的内容。在本例中为text/html

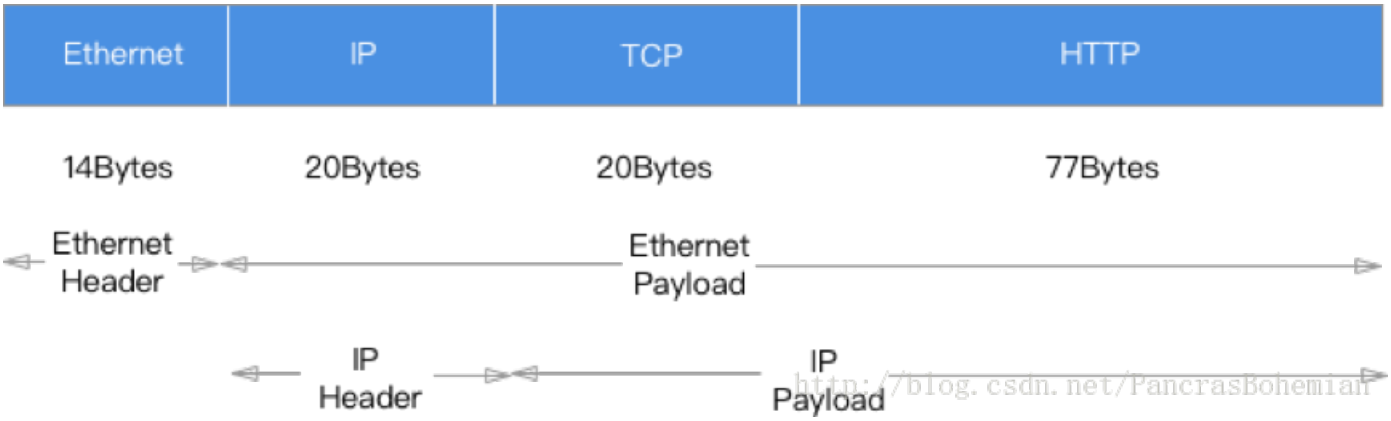
Step3 Packet Structure

要求画一下GET包的结构，我们可以看到





随着你鼠标滑过中间那些不同的协议，下面会有不同的内容被高亮，依据这个，我们可以画出图来。
而大小可以在程序的最下面找到
画出Ethernet的头部和有效载荷（payload，指除去协议首部之外实际传输的数据）



这就是我画的

Step4 Protocol Overhead

Protocol Overhead指的是协议开销，由于协议本身的报头等内容也需要占用一定的空间，用来标识该种协议、报文内各个字段的含义等信息，这种内容就是协议开销了。
下载的包从一个Info中带有SYN，ACK信号的包开始，到下面第一个遇到的HTTP包后面的TCP包为止。

其中协议开销一共有78+54+54+60 = 246字节
HTTP有效信息一共有77字节，占23.84%可以说协议开销占比较大。也较为重要，因为标识各个字段的含义等信息是必不可少的。不过要是开销能小一些会更好

Step5 Demultiplexing Keys

分别对应0x08 00(16进制) 0x6（十六进制）
Which Ethernet header field is the demultiplexing key that tells it the next higher layer is IP? What value is used in this field to indicate “IP”?
上面红色框，对应的是0x08 00
Which IP header field is the demultiplexing key that tells it the next higher layer is TCP? What value is used in this field to indicate “TCP”?
下面红色框对应的是0x6