LAB-PROTOCOL-LAYERS

PREPARE

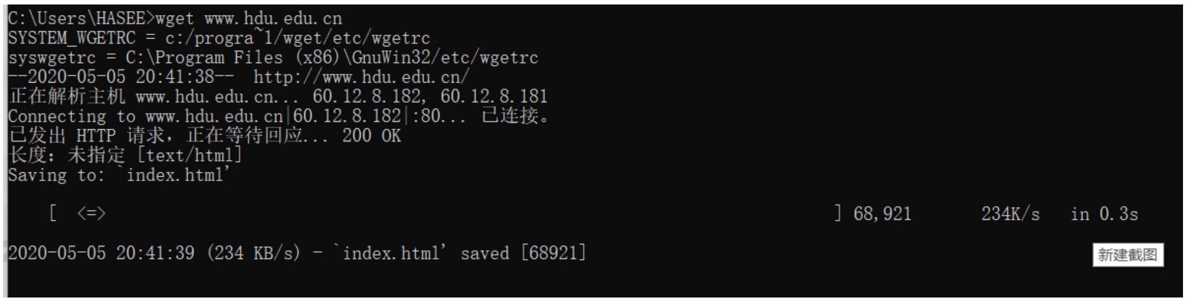
下载wget安装包

并且设置好相应的系统变量。

Step 1: Capture a Trace

1 选用网页：http://www.hdu.edu.cn/index.php 航电的主页

用命令：`wget www.hdu.edu.cn`

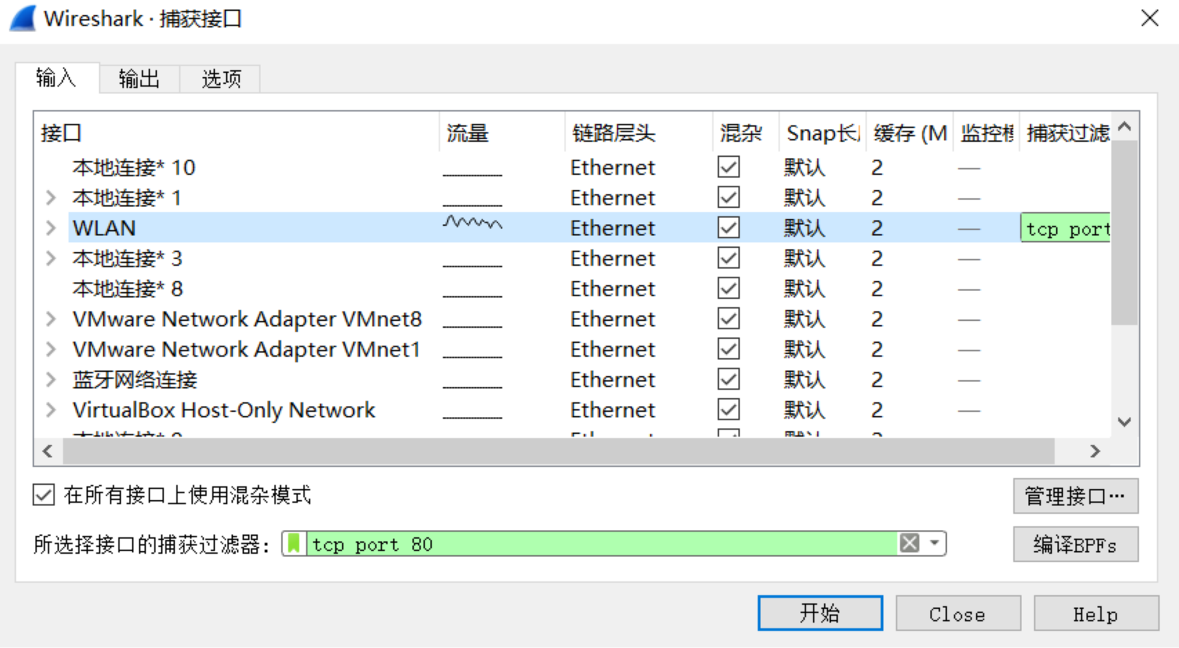


```
C:\Users\HASEE>wget www.hdu.edu.cn
SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = C:\Program Files (x86)\GnuWin32/etc/wgetrc
--2020-05-05 20:41:38--  http://www.hdu.edu.cn/
正在解析主机 www.hdu.edu.cn... 60.12.8.182, 60.12.8.181
Connecting to www.hdu.edu.cn|60.12.8.182|:80... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：未指定 [text/html]
Saving to: `index.html'

    [ <=>                                              ] 68,921      234K/s   in 0.3s

2020-05-05 20:41:39 (234 KB/s) - `index.html' saved [68921]
```

捕获得到200 OK。

2 关闭无关程序

3 启动wire shark 并且设置filter



接口	流量	链路层头	混杂	Snap长	缓存 (M	监控模	捕获过滤
本地连接* 10		Ethernet	☑	默认	2	—	
本地连接* 1		Ethernet	☑	默认	2	—	
WLAN		Ethernet	☑	默认	2	—	tcp port
本地连接* 3		Ethernet	☑	默认	2	—	
本地连接* 8		Ethernet	☑	默认	2	—	
VMware Network Adapter VMnet8		Ethernet	☑	默认	2	—	
VMware Network Adapter VMnet1		Ethernet	☑	默认	2	—	
蓝牙网络连接		Ethernet	☑	默认	2	—	
VirtualBox Host-Only Network		Ethernet	☑	默认	2	—	

☑ 在所有接口上使用混杂模式

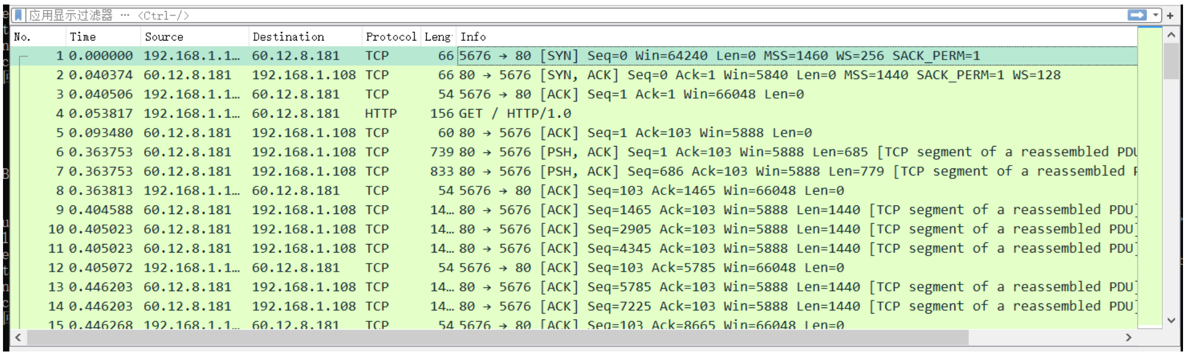
所选择接口的捕获过滤器：tcp port 80

电脑用的是wlan连接网络。

4 捕获

在启动wire shark捕获后，再次运行命令`wget www.hdu.edu.cn`

5 捕获结果

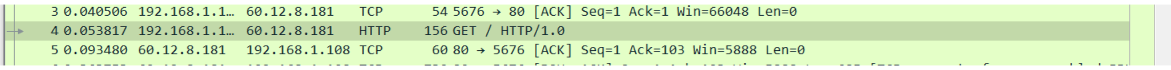


```
应用显示过滤器 … <Ctrl-/>
No.  Time      Source        Destination   Protocol Leng Info
   1 0.000000  192.168.1.1…  60.12.8.181   TCP       66 5676 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   2 0.040374  60.12.8.181   192.168.1.108 TCP       66 80 → 5676 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1440 SACK_PERM=1 WS=128
   3 0.040506  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
   4 0.053817  192.168.1.1…  60.12.8.181   HTTP     156 GET / HTTP/1.0
   5 0.093480  60.12.8.181   192.168.1.108 TCP       60 80 → 5676 [ACK] Seq=1 Ack=103 Win=5888 Len=0
   6 0.363753  60.12.8.181   192.168.1.108 TCP      739 80 → 5676 [PSH, ACK] Seq=1 Ack=103 Win=5888 Len=685 [TCP segment of a reassembled PDU
   7 0.363753  60.12.8.181   192.168.1.108 TCP      833 80 → 5676 [PSH, ACK] Seq=686 Ack=103 Win=5888 Len=779 [TCP segment of a reassembled F
   8 0.363813  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=103 Ack=1465 Win=66048 Len=0
   9 0.404588  60.12.8.181   192.168.1.108 TCP      14… 80 → 5676 [ACK] Seq=1465 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
  10 0.405023  60.12.8.181   192.168.1.108 TCP      14… 80 → 5676 [ACK] Seq=2905 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
  11 0.405023  60.12.8.181   192.168.1.108 TCP      14… 80 → 5676 [ACK] Seq=4345 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
  12 0.405072  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=103 Ack=5785 Win=66048 Len=0
  13 0.446203  60.12.8.181   192.168.1.108 TCP      14… 80 → 5676 [ACK] Seq=5785 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
  14 0.446203  60.12.8.181   192.168.1.108 TCP      14… 80 → 5676 [ACK] Seq=7225 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
  15 0.446268  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=103 Ack=8665 Win=66048 Len=0
```

其中60.12.8.181就是www.hdu.edu.cn 对应的一个ip，共捕获了77个包（分组数据）

Step 2: Inspect the Trace

1 get

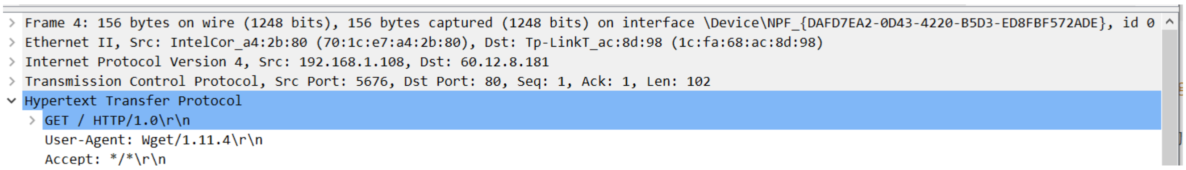


```
   3 0.040506  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
   4 0.053817  192.168.1.1…  60.12.8.181   HTTP     156 GET / HTTP/1.0
   5 0.093480  60.12.8.181   192.168.1.108 TCP       60 80 → 5676 [ACK] Seq=1 Ack=103 Win=5888 Len=0
```

针对第四个包进行查看，该包使用了HTTP协议。

协议栈

它对应的协议栈如下：



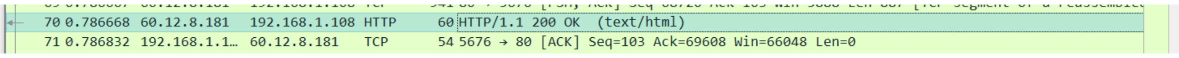
```
> Frame 4: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface \Device\NPF_{DAFD7EA2-0D43-4220-B5D3-ED8FBF572ADE}, id 0
> Ethernet II, Src: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80), Dst: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98)
> Internet Protocol Version 4, Src: 192.168.1.108, Dst: 60.12.8.181
> Transmission Control Protocol, Src Port: 5676, Dst Port: 80, Seq: 1, Ack: 1, Len: 102
v Hypertext Transfer Protocol
  > GET / HTTP/1.0\r\n
    User-Agent: Wget/1.11.4\r\n
    Accept: */*\r\n
```

从上到下依次是: 帧, 以太2, IPV4, TCP, HTTP.

它们在包内的顺序也是从前到后的.

不同的协议占的位数不一样.

2 回复

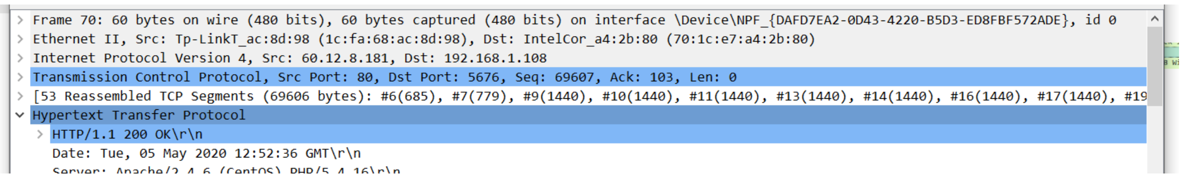


```
  70 0.786668  60.12.8.181   192.168.1.108 HTTP      60 HTTP/1.1 200 OK  (text/html)
  71 0.786832  192.168.1.1…  60.12.8.181   TCP       54 5676 → 80 [ACK] Seq=103 Ack=69608 Win=66048 Len=0
```

回复在第70个包中,含有200 OK.

协议栈

它对应的协议栈如下:



```
> Frame 70: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{DAFD7EA2-0D43-4220-B5D3-ED8FBF572ADE}, id 0
> Ethernet II, Src: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98), Dst: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80)
> Internet Protocol Version 4, Src: 60.12.8.181, Dst: 192.168.1.108
> Transmission Control Protocol, Src Port: 80, Dst Port: 5676, Seq: 69607, Ack: 103, Len: 0
> [53 Reassembled TCP Segments (69606 bytes): #6(685), #7(779), #9(1440), #10(1440), #11(1440), #13(1440), #14(1440), #16(1440), #17(1440), #19
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Tue, 05 May 2020 12:52:36 GMT\r\n
    Server: Apache/2.4.6 (CentOS) PHP/5.4.16\r\n
```

Step 3: Packet Structure

GET包每个协议占的字节数

Frame 4:	156字节(不是某一个协议,而是这个帧的总大小)
Ethernet 2, Src:	14字节
IPV4:	20字节
TCP:	20字节
HTTP:	102字节

图就不画了. 协议间依次排列. 更底层的协议更靠前.

回复包每个协议占的字节数

Frame 70:	60字节
Ethernet 2, Src:	20字节(前14个字节+包的最后6个字节,后面的全是0)
IPV4:	20字节
TCP:	20字节

53 Reassembled TCP Segments: 69606字节(不是这个包里的,是其他包的连接信息) 它们一块构成了HTTP协议(分散在其他包里)

Step 4: Protocol Overhead

估计假设

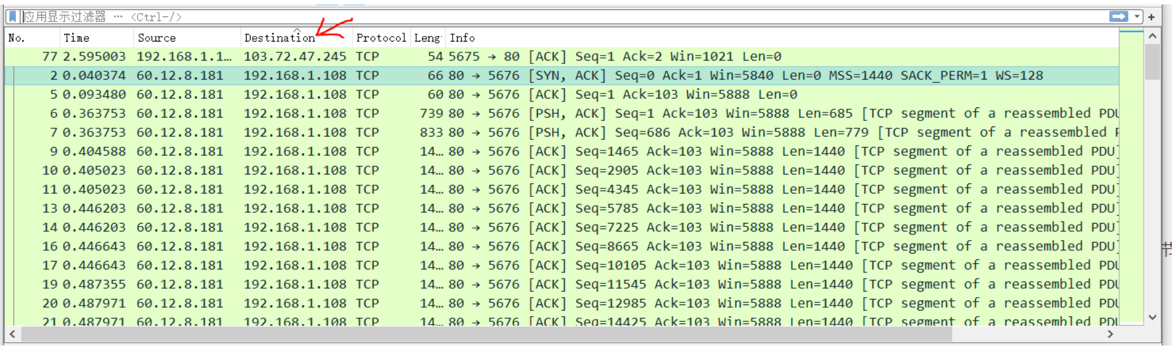
估计每个包的Ethernet 2,Src 协议占的都是14字节(有少量占20字节,还是都算作14字节)

估计每个包的IPV4协议占20字节.

估计每个包的TCP协议占20字节.

下载包的数量

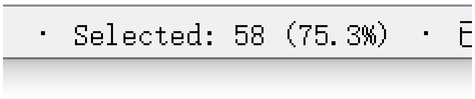
对Destination进行排序:



No.	Time	Source	Destination	Protocol	Leng	Info
77	2.595003	192.168.1.1…	103.72.47.245	TCP	54	5675 → 80 [ACK] Seq=1 Ack=2 Win=1021 Len=0
2	0.040374	60.12.8.181	192.168.1.108	TCP	66	80 → 5676 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1440 SACK_PERM=1 WS=128
5	0.093480	60.12.8.181	192.168.1.108	TCP	60	80 → 5676 [ACK] Seq=1 Ack=103 Win=5888 Len=0
6	0.363753	60.12.8.181	192.168.1.108	TCP	739	80 → 5676 [PSH, ACK] Seq=1 Ack=103 Win=5888 Len=685 [TCP segment of a reassembled PDU
7	0.363753	60.12.8.181	192.168.1.108	TCP	833	80 → 5676 [PSH, ACK] Seq=686 Ack=103 Win=5888 Len=779 [TCP segment of a reassembled P
9	0.404588	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=1465 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
10	0.405023	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=2905 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
11	0.405023	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=4345 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
13	0.446203	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=5785 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
14	0.446203	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=7225 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
16	0.446643	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=8665 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU]
17	0.446643	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=10105 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU
19	0.487355	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=11545 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU
20	0.487971	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=12985 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU
21	0.487971	60.12.8.181	192.168.1.108	TCP	14…	80 → 5676 [ACK] Seq=14425 Ack=103 Win=5888 Len=1440 [TCP segment of a reassembled PDU

找到所有Destination=192.168.1.108的包 (都是收到的包)

一共收到有58个包:


· Selected: 58 (75.3%) · E

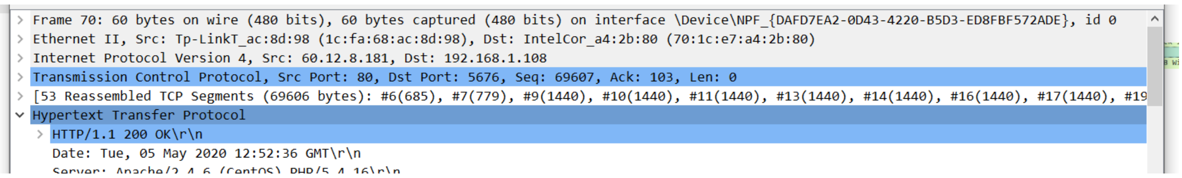
估计

这58个包 共有Ethernet + IPV4 + TCP协议的估计字节= 58*(14+20+20)=3132字节

所以对应的开销大约是3132字节.

有效数据

再次查看刚才的回复的包的协议栈:



```
> Frame 70: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{DAFD7EA2-0D43-4220-B5D3-ED8FBF572ADE}, id 0
> Ethernet II, Src: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98), Dst: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80)
> Internet Protocol Version 4, Src: 60.12.8.181, Dst: 192.168.1.108
> Transmission Control Protocol, Src Port: 80, Dst Port: 5676, Seq: 69607, Ack: 103, Len: 0
> [53 Reassembled TCP Segments (69606 bytes): #6(685), #7(779), #9(1440), #10(1440), #11(1440), #13(1440), #14(1440), #16(1440), #17(1440), #19
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Tue, 05 May 2020 12:52:36 GMT\r\n
    Server: Apache/2.4.6 (CentOS) PHP/5.4.16\r\n
```

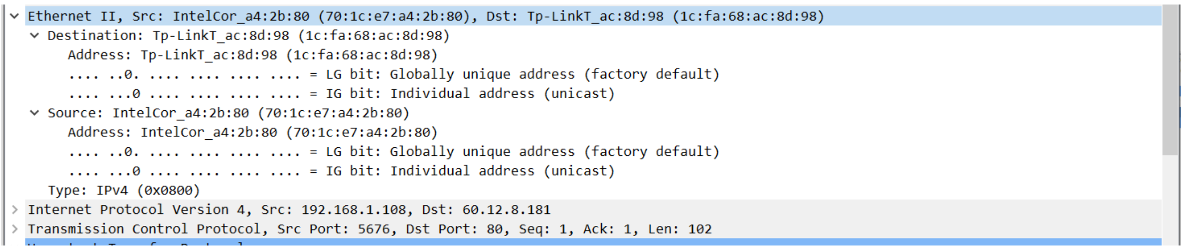
它给出了在其他包中 下载的有效数据总量(69606字节)

Step 5: Demultiplexing Keys

基本上任意一个包都用的是以太+IPv4+TCP的协议栈

1 以太协议包涵IPv4

打开一个包的以太协议:

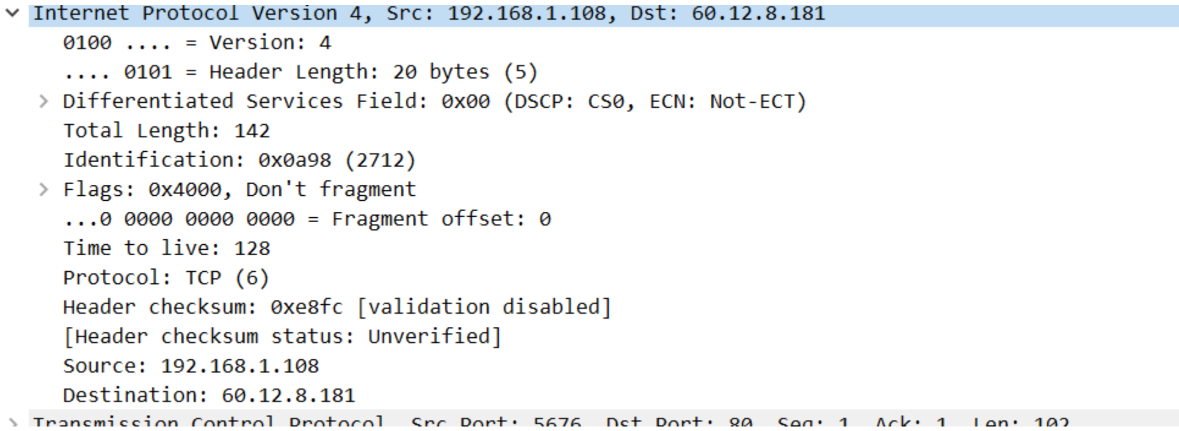


```
v Ethernet II, Src: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80), Dst: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98)
  v Destination: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98)
      Address: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80)
      Address: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.108, Dst: 60.12.8.181
> Transmission Control Protocol, Src Port: 5676, Dst Port: 80, Seq: 1, Ack: 1, Len: 102
```

可以看到,它有个字段是Type,然后值是IPv4(0x0800),所以应该就是这个字段是多路分解键.

2 IP协议包涵TCP

打开一个包的IP协议:



```
v Internet Protocol Version 4, Src: 192.168.1.108, Dst: 60.12.8.181
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 142
    Identification: 0x0a98 (2712)
  > Flags: 0x4000, Don't fragment
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0xe8fc [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.108
    Destination: 60.12.8.181
> Transmission Control Protocol, Src Port: 5676, Dst Port: 80, Seq: 1, Ack: 1, Len: 102
```

它有个字段叫做Protocol, 值是TCP(0x06), 所以应该是这个字段是多路分解键.