LAB-TCP

PREPARE

下载wget安装帧

并且设置好相应的系统变量。

Step 1: Capture a Trace

1 选用网页: www.hdu.edu.cn/asset/home/images/logo.png hdu的logo图片

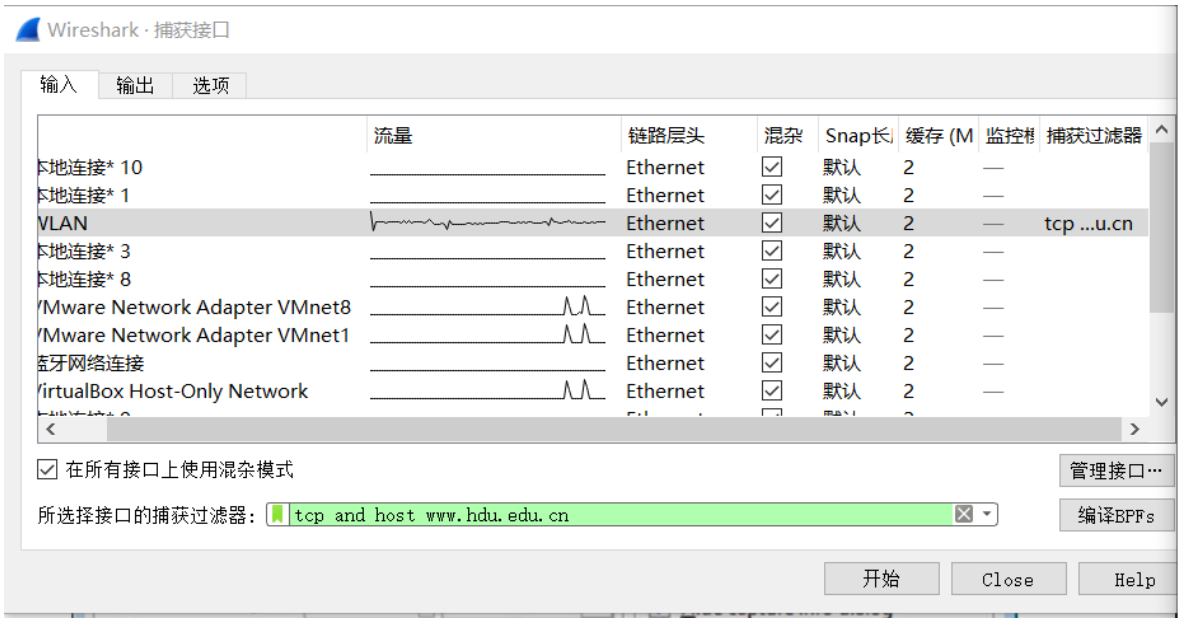
用命令：wget www.hdu.edu.cn/asset/home/images/logo.png

```
C:\Users\HASEE>wget www.hdu.edu.cn/asset/home/images/logo.png
SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = C:\Program Files (x86)\GnuWin32/etc/wgetrc
--2020-06-19 00:54:36--  http://www.hdu.edu.cn/asset/home/images/logo.png
正在解析主机 www.hdu.edu.cn... 60.12.8.181, 60.12.8.182
Connecting to www.hdu.edu.cn|60.12.8.181|:80... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：337187 (329K) [image/png]
Saving to: `logo.png.1'

100%[======================================================================================>] 337,187

2020-06-19 00:54:40 (82.4 KB/s) - `logo.png.1' saved [337187/337187]
```

捕获得到200 OK。

2 关闭无关程序

3 启动wire shark 并且设置filter：tcp and host www.hdu.edu.cn



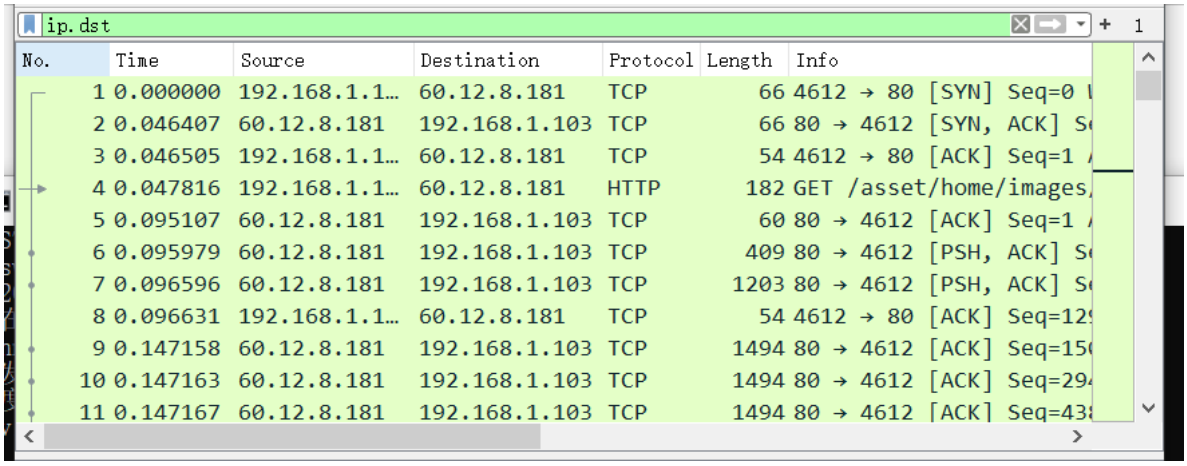
电脑用的是wlan连接网络。

4 捕获

在启动wire shark捕获后，再次运行命令

```
wget www.hdu.edu.cn/asset/home/images/logo.png
```

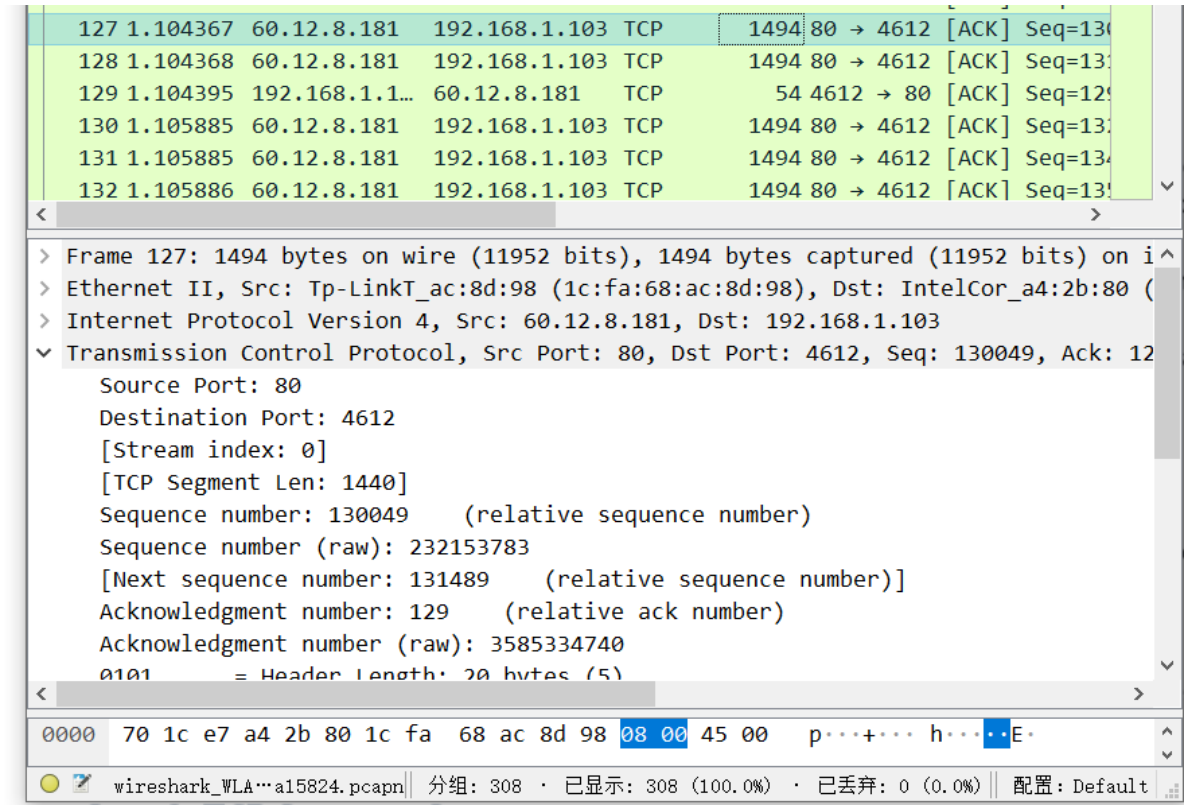
5 捕获结果



其中60.12.8.181就是www.hdu.edu.cn 对应的一个ip，共捕获了308个帧

Step 2: Inspect the Trace

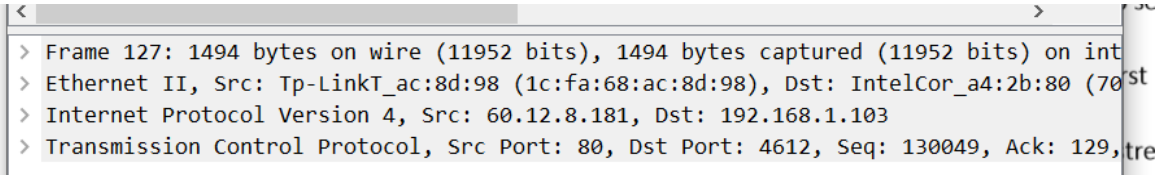
中间的一个tcp帧：



针对第四个帧进行查看，该帧使用了TCP协议。

协议栈

它对应的协议栈如下：



从上到下依次是: 帧, 以太2, IPV4, TCP.

它们在帧内的顺序也是从前到后的.

不同的协议占的位数不一样.

它的TCP协议的具体字段：

```
> Frame 127: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on int
> Ethernet II, Src: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98), Dst: IntelCor_a4:2b:80 (70
> Internet Protocol Version 4, Src: 60.12.8.181, Dst: 192.168.1.103
v Transmission Control Protocol, Src Port: 80, Dst Port: 4612, Seq: 130049, Ack: 129,
    Source Port: 80
    Destination Port: 4612
    [Stream index: 0]
    [TCP Segment Len: 1440]
    Sequence number: 130049    (relative sequence number)
    Sequence number (raw): 232153783
    [Next sequence number: 131489    (relative sequence number)]
    Acknowledgment number: 129    (relative ack number)
    Acknowledgment number (raw): 3585334740
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 54
    [Calculated window size: 6912]
    [Window size scaling factor: 128]
    Checksum: 0x89ce [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (1440 bytes)
    [Reassembled PDU in frame: 306]
    TCP segment data (1440 bytes)
```

可以看到：

源端口：80 一般涉及网页访问的就用80端口号

目标端口：4612 hdu的服务器端口号

序列号：130049， 指明当前帧的序列号

ACK号：129

头长度：20 B

flags：

```
v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A····]
```

窗口大小：54

校验和：0x89ce

紧急指针：0

Step 3: TCP Segment Structure

TCP帧中TCP协议每个字段占的字节数：

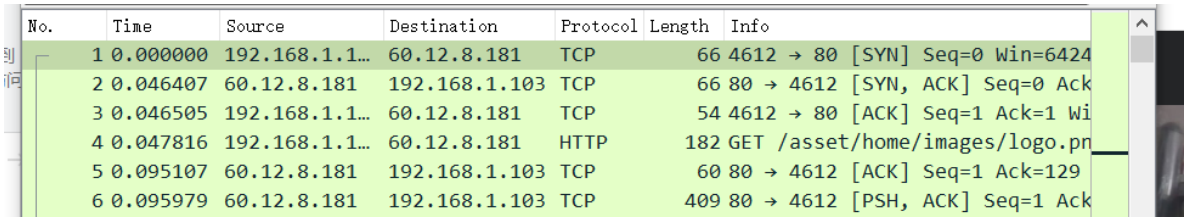
字段名称	占的大小（字节）
源端口号	2
目标端口号	2
序列号	4
ack号	4
头长度	0.5
Flags	1.5
窗口大小	2
校验和	2
紧急指针	2

图就不画了. 字段间依次排列.

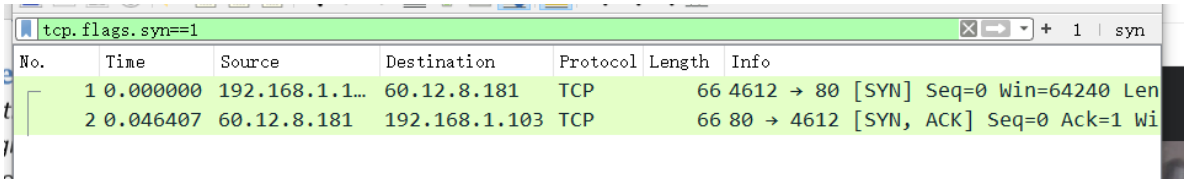
Step 4: TCP Connection Setup/Teardown

三次握手

往前找，前两个帧恰好是SYN标志的：



应用过滤器：



就只有前两个是syn

先看看每个帧的字段数值吧：

第一个帧：

```
Transmission Control Protocol, Src Port: 4612, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 4612
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 3585334611
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
```

Seq = 0 , Flag = SYN ， Time = 0.000000

第二个帧:

```
    2 0.046407  60.12.8.181   192.168.1.103 TCP        66 80 → 4612 [SYN, ACK] Seq=0 Ack
    3 0.046505  192.168.1.1…  60.12.8.181   TCP        54 4612 → 80 [ACK] Seq=1 Ack=1 Wi
> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF
> Ethernet II, Src: Tp-LinkT_ac:8d:98 (1c:fa:68:ac:8d:98), Dst: IntelCor_a4:2b:80 (70:1c:e7:a
> Internet Protocol Version 4, Src: 60.12.8.181, Dst: 192.168.1.103
Transmission Control Protocol, Src Port: 80, Dst Port: 4612, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 4612
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 232023734
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 3585334612
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
```

Seq = 0 , Flag = (SYN， ACK) , ack = 1 ， Time = 0.046407

第三个帧:

```
    3 0.046505  192.168.1.1…  60.12.8.181   TCP        54 4612 → 80 [ACK] Seq=1 Ack=1 Wi
    4 0.047816  192.168.1.1…  60.12.8.181   HTTP      182 GET /asset/home/images/logo.pn
> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF
> Ethernet II, Src: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80), Dst: Tp-LinkT_ac:8d:98 (1c:fa:68:a
> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 60.12.8.181
Transmission Control Protocol, Src Port: 4612, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 4612
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 3585334612
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 232023735
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
```

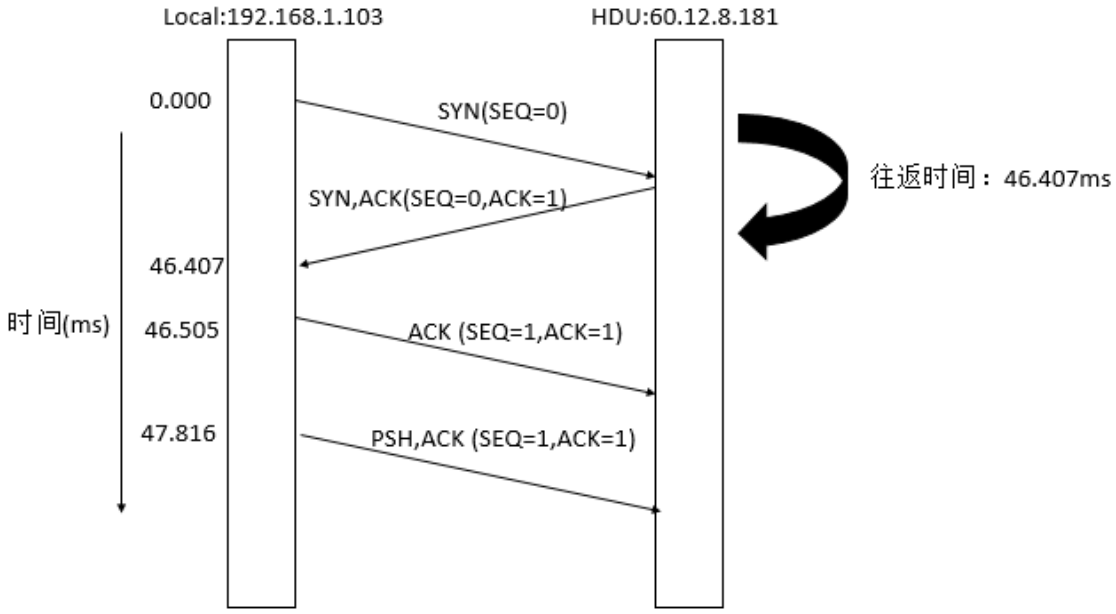
Seq = 1 , Flag = (ACK) , ack = 1 ， Time = 0.046505

第四个帧: （GET帧）

```
    4 0.047816  192.168.1.1…  60.12.8.181    HTTP      182 GET /asset/home/images/logo.pn
> Frame 4: 182 bytes on wire (1456 bits), 182 bytes captured (1456 bits) on interface \Dev
> Ethernet II, Src: IntelCor_a4:2b:80 (70:1c:e7:a4:2b:80), Dst: Tp-LinkT_ac:8d:98 (1c:fa:6
> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 60.12.8.181
v Transmission Control Protocol, Src Port: 4612, Dst Port: 80, Seq: 1, Ack: 1, Len: 128
    Source Port: 4612
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 128]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 3585334612
    [Next sequence number: 129    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 232023735
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
```

Seq = 1 , Flag = (PSH, ACK) , ack = 1 ， Time = 0.047816

图:



SYN携带的参数

查看第一个SYN携带的额外参数:

```
v Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operat
  > TCP Option - Maximum segment size: 1460 bytes
  > TCP Option - No-Operation (NOP)
  > TCP Option - Window scale: 8 (multiply by 256)
  > TCP Option - No-Operation (NOP)
  > TCP Option - No-Operation (NOP)
  > TCP Option - SACK permitted
> [Timestamps]
```

可以看到有额外的参数:

MSS 最大帧大小: 1460

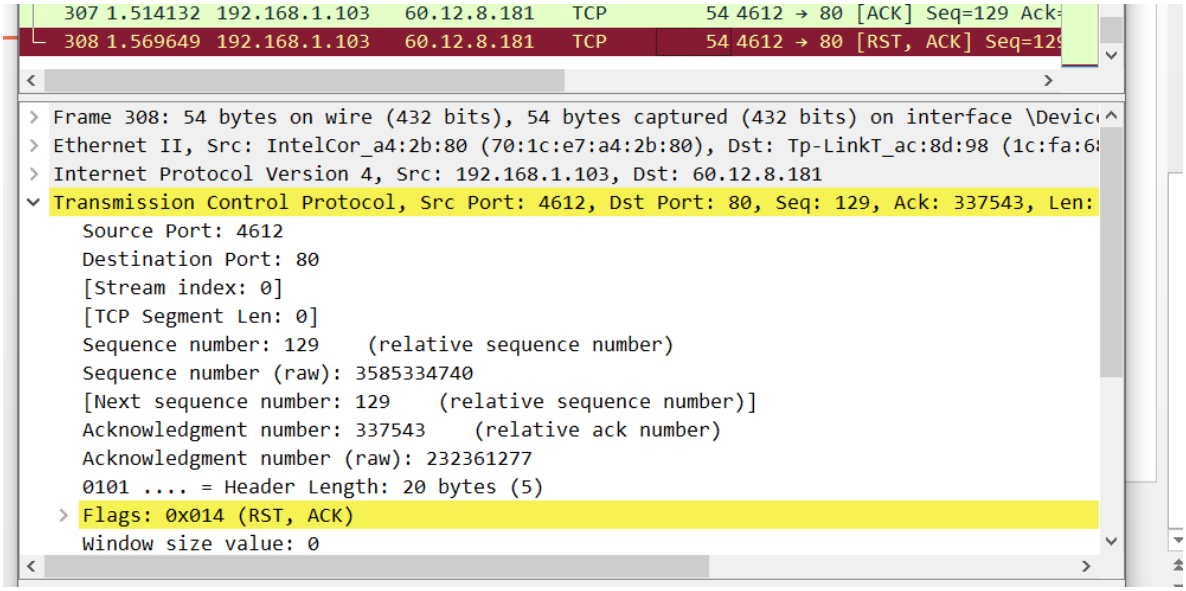
Window Scale: 8 窗口缩放尺度，以后的我方传递给服务器的窗口大小都必须乘以2^8（256）

SACK permitted：确认可以开启sack功能

没有timesamps(长度为0)

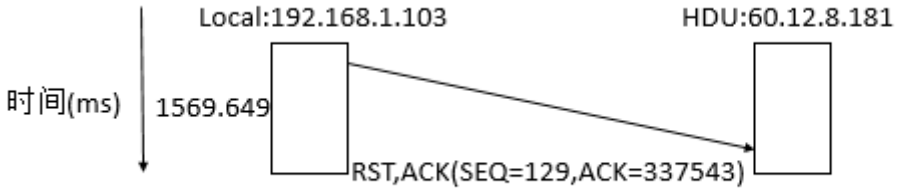
FIN/RST Teardown

最后一个帧就是发出去的RST帧。



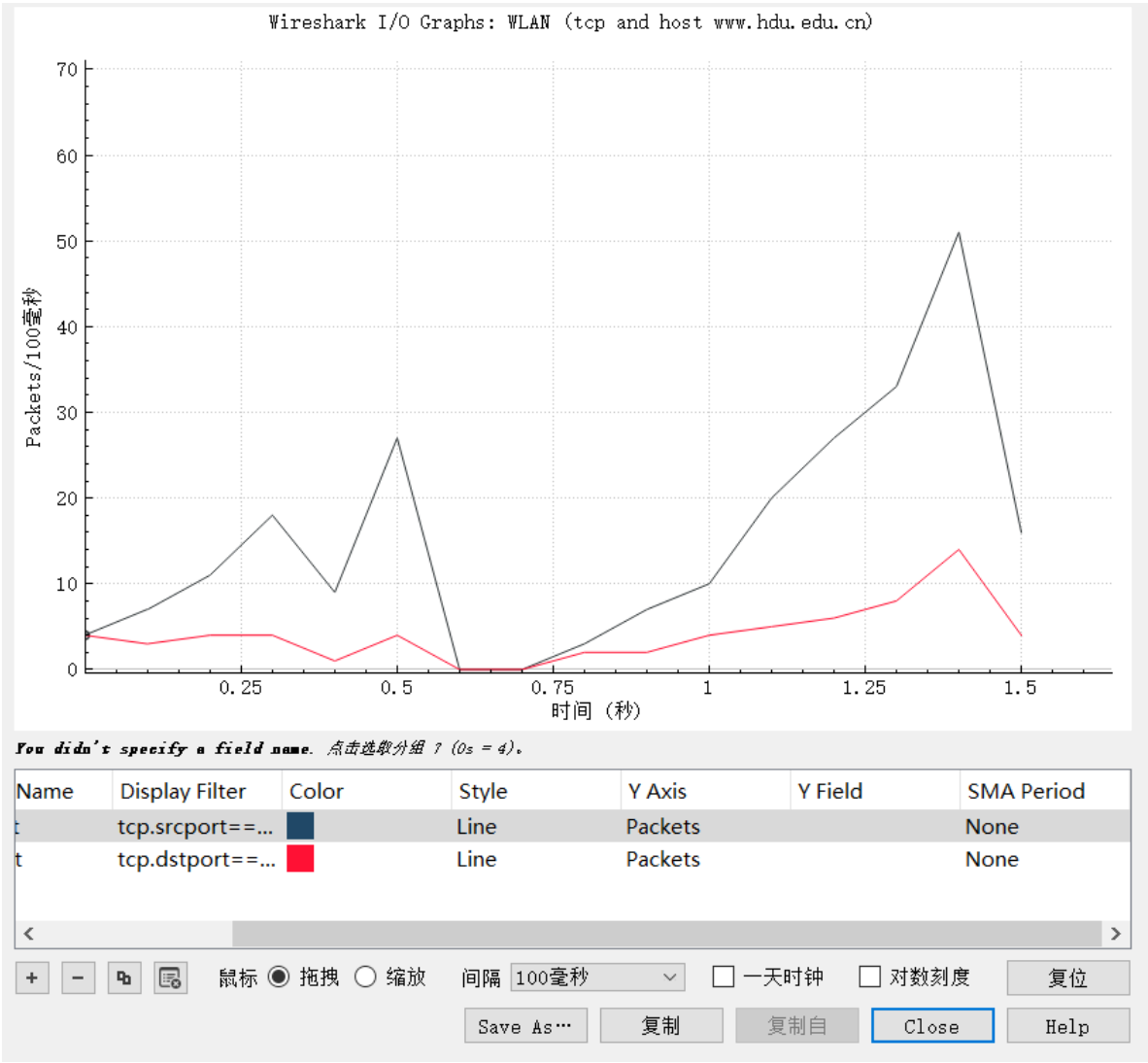
我的是RST，不需要对方回复了。

图：



Step 5: TCP Data Transfer

图像：



感觉这个图片不是很大，我估计是因为下载的东西太少了，还没到那个最大速率的时候，就已经下完了。我重新找了一个大的图片来下载：

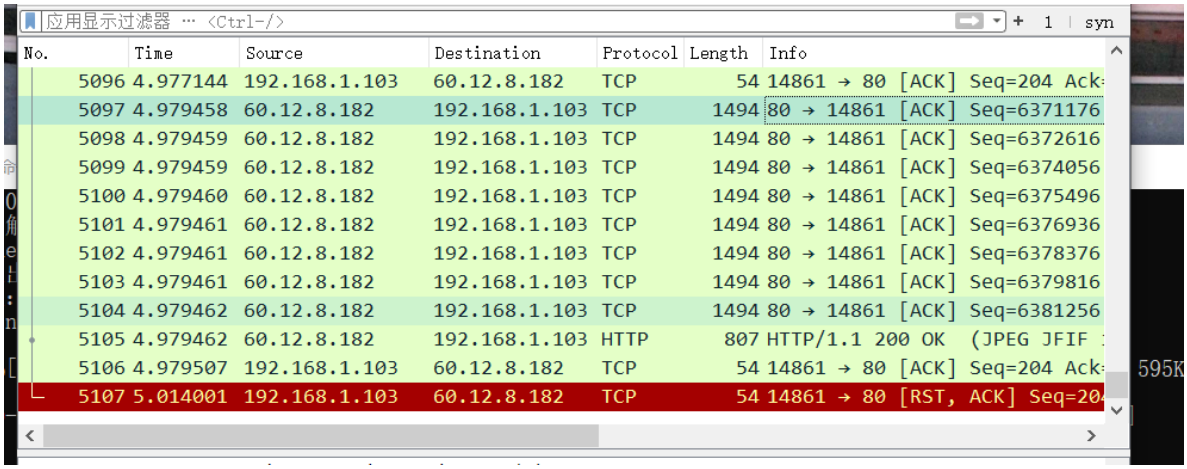
地址：http://dangan.hdu.edu.cn/_upload/article/images/5d/2d/ad3347084e55987b4d5873d17428/48beb93c-7b9e-430e-bb74-bd6b68cf1cd7.jpg

wget:

```
C:\Users\HASEE>wget http://dangan.hdu.edu.cn/_upload/article/images/5d/2d/ad3347084e55987b4d5873d17428/48beb93c-7b9e-430e-bb74-bd6b68cf1cd7.jpg
SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = C:\Program Files (x86)\GnuWin32/etc/wgetrc
--2020-06-19 02:26:30--  http://dangan.hdu.edu.cn/_upload/article/images/5d/2d/ad3347084e55987b4d5873d17428/48beb93c-7b9e-430e-bb74-bd6b68cf1cd7.jpg
正在解析主机 dangan.hdu.edu.cn... 60.12.8.181, 60.12.8.182
Connecting to dangan.hdu.edu.cn|60.12.8.181|:80... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：6383059 (6.1M) [image/jpeg]
Saving to: `48beb93c-7b9e-430e-bb74-bd6b68cf1cd7.jpg'

100%[======================================>] 6,383,059    595K/s   in 12s

2020-06-19 02:26:44 (529 KB/s) - `48beb93c-7b9e-430e-bb74-bd6b68cf1cd7.jpg' saved [6383059/6383059]
```

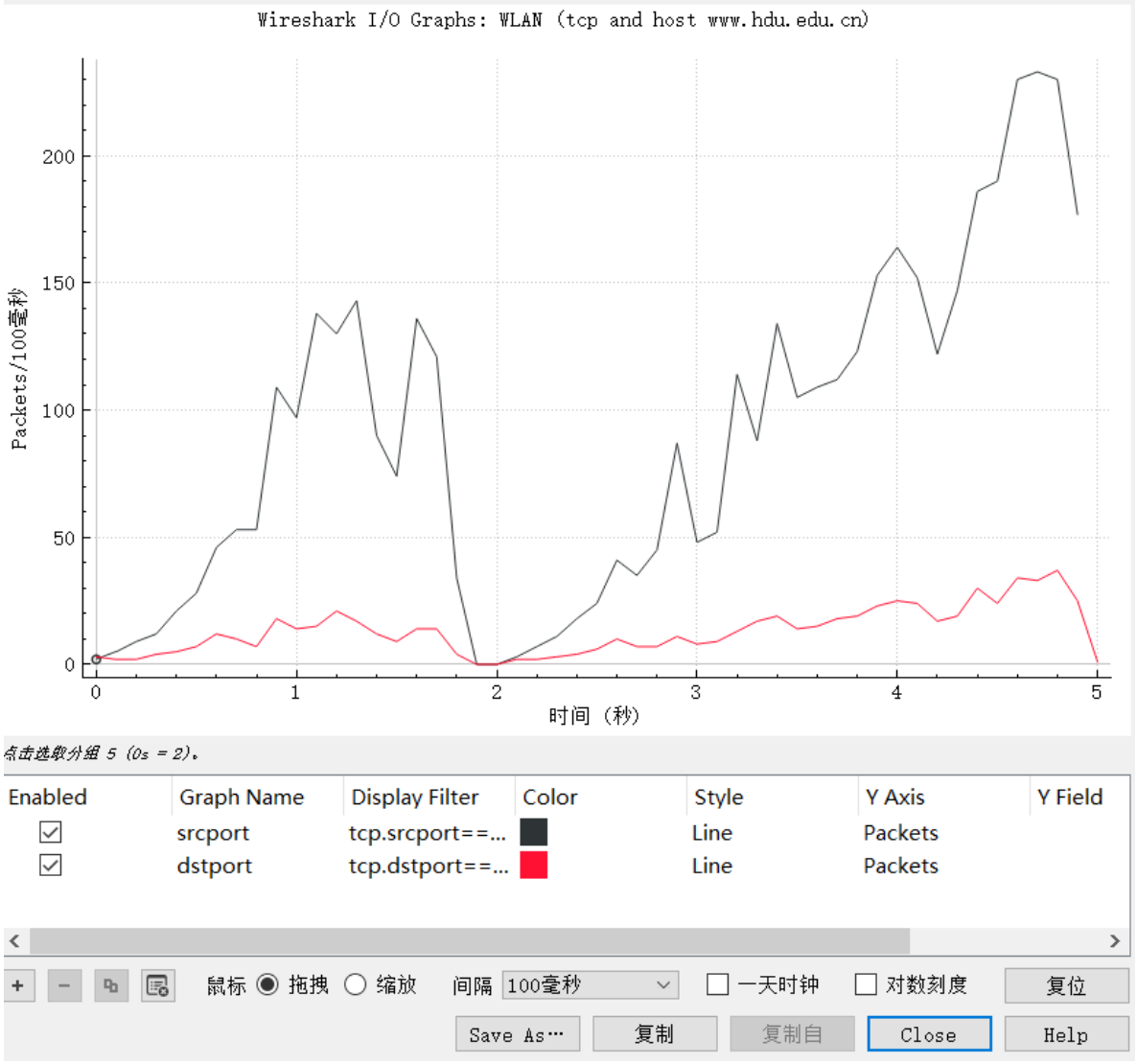
可以看到该图片大小为6MB左右。 ip: 60.12.8.181

wireshark捕捉：

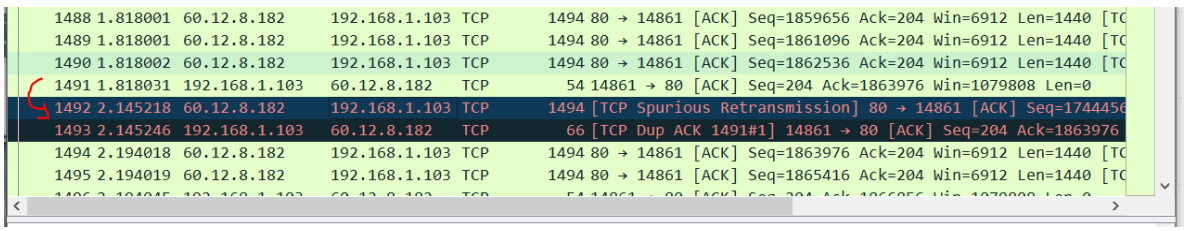


捕了5107帧

画图：



发现途中2s左右断了一下，查看该处附近的包：



发现出现了一个 spurious retransmission 虚假重传，网上百度一下

tcp 抓包出现spurious retransmission

一、tcp虚假重传

指实际上并没有超时，但看起来超时了，导致虚假超时重传的原因有很多种：

（1）对于部分移动网络，当网络发生切换时会导致网络延时突增

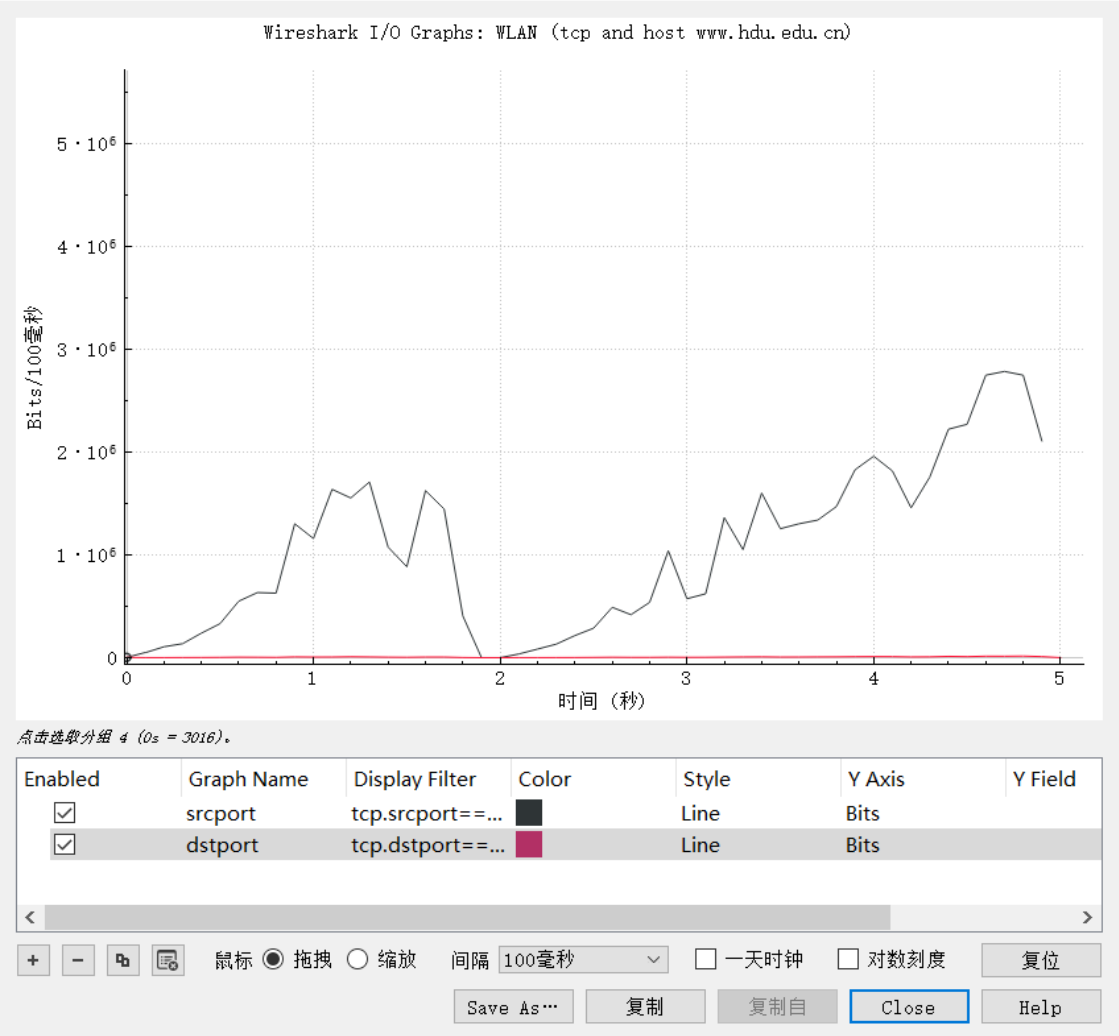
（2）当网络的可用带宽突然变小时，网络rtt会出现突增的情况，这会导致虚假超时重传

（3）网络丢包（原始和重传的包都有可能丢包）会导致虚假重传超时。

很奇怪。

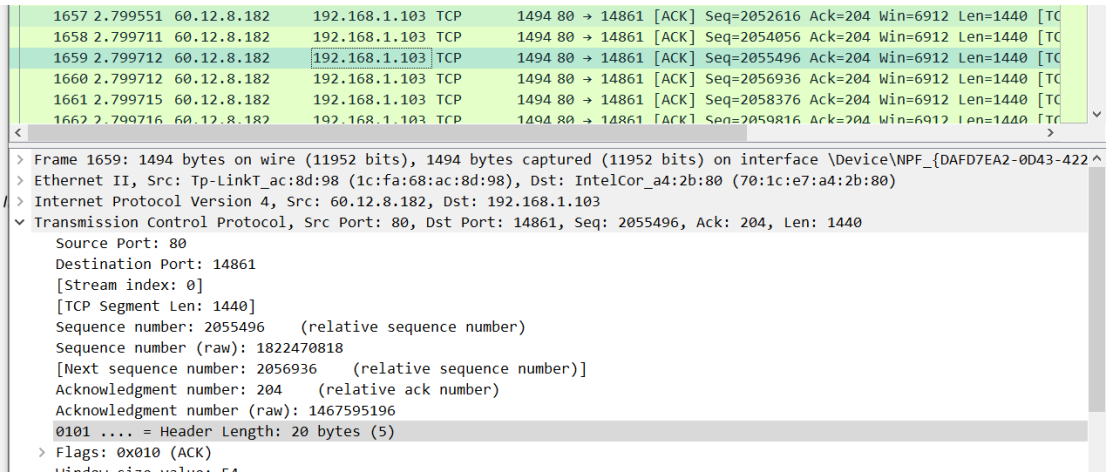
问题：

1. 大概速度： 看不出来上限，最大值是2600包/秒



2. 8e7 位/秒

2.

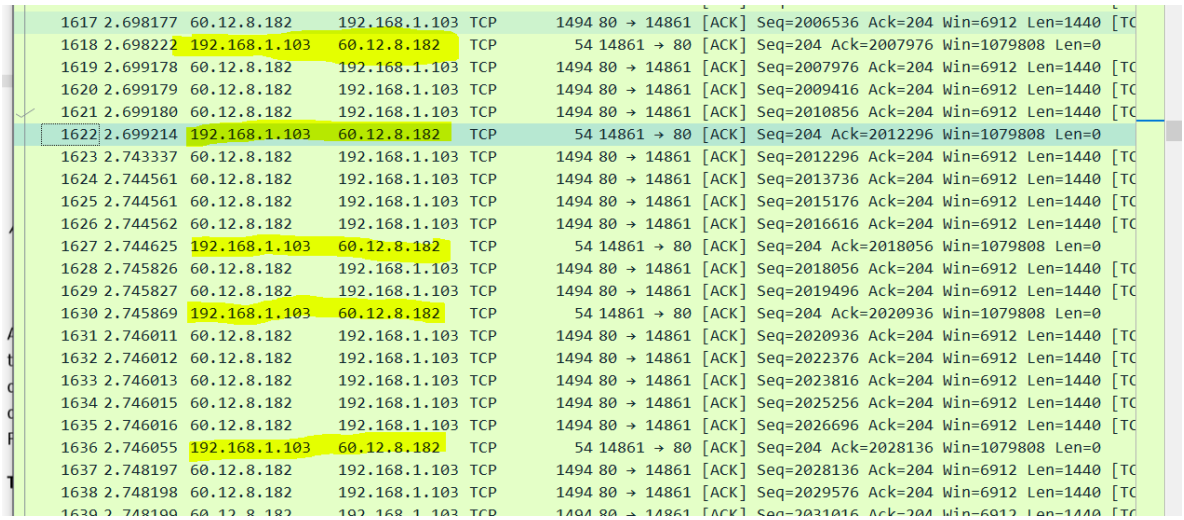


这里的第1659帧，总大小1494，TCP占20，所以占用比率是20/1494 = 1.34%

3. 上传的速率小得多：大概是250包/秒， 1e5位/秒

4. x+1

后面截一个 1个上传确认帧 对应多个下载帧的图片：



可以看到黄色高亮的上传帧之间隔了好几个 下载的帧。