实验报告：protocol layer

课程名称：计算机网络	**年级：**2023 级	**实践成绩：**
指导教师：章玥	**姓名：**张建夫	
实践名称：protocol layer	**学号：**10235101477	**实践日期：**2024/11/11
实践编号：	**组号：**	**实践时间：**8:00~9:30

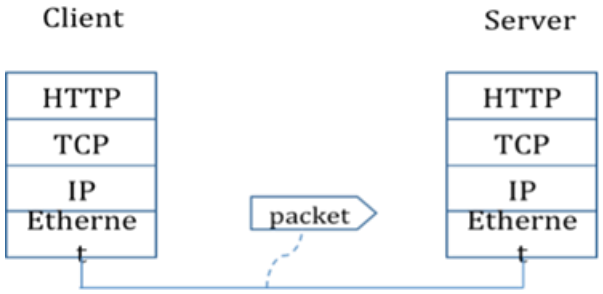
一、目的

1. 学习协议和分层如何用数据包表示;
2. 熟悉 wireshark 软件、curl、wget 等常用软件的使用，掌握网络抓包的方法，能在 所用电脑上进 行抓包;
3. 了解 IP 数据包格式，能应用该软件分析数据包格式，查看抓到的包的内容，并分析对应的 IP 数据 包格式;
4. 抓包分析数据包，估算协议的开销;
5. 通过数据包抓取实验，将理论与实践相结合，深入理解协议层的字段与结构特征.

二、实验内容与实验步骤

1.本次实验的实验内容与背景：

利用 wireshark 抓取应用层 http 协议 get 请求的数据包，深入了解 tcp 传输层协议的三次握手与断开连接的四次挥手，同时验证 http 协议 get 请求的数据包的协议栈是否如下图所示：



并详细列出各个字段所占字节的大小以及计算总开销。

2.实验步骤：

1）利用 wireshark 捕获本机与目标网址的服务器（自己选择网址）之间发送的数据包（数据包的发送使用 wget 实现）

2）对得到的数据包进行分析，判断出哪些数据包是握手过程，哪些是挥手过程

3）对 http 协议的 get 请求数据包的结构进行分析（各个协议所占的大小和头部的比例等）

三、实验环境

调用 dxdiag 工具：

Operating System: Windows 11 家庭中文版 64-bit (10.0, Build 22621) (22621.ni_release.220506-1250)
Language: Chinese (Simplified) (Regional Setting: Chinese (Simplified))
System Manufacturer: HP

System Model: HP Pavilion Aero Laptop 13-be2xxx
BIOS: F.13 (type: UEFI)
Processor: AMD Ryzen 5 7535U with Radeon Graphics (12 CPUs), ~2.9GHz
Memory: 16384MB RAM
Available OS Memory: 15574MB RAM
Page File: 27604MB used, 5685MB available
Windows Dir: C:\WINDOWS
DirectX Version: DirectX 12
DX Setup Parameters: Not found
User DPI Setting: 144 DPI (150 percent)
System DPI Setting: 192 DPI (200 percent)
DWM DPI Scaling: UnKnown
Miracast: Available, with HDCP
Microsoft Graphics Hybrid: Not Supported

四、实验过程与分析

实验前要选择一个网址作为抓包对象，我选择的是 www.ecnu.edu.cn.

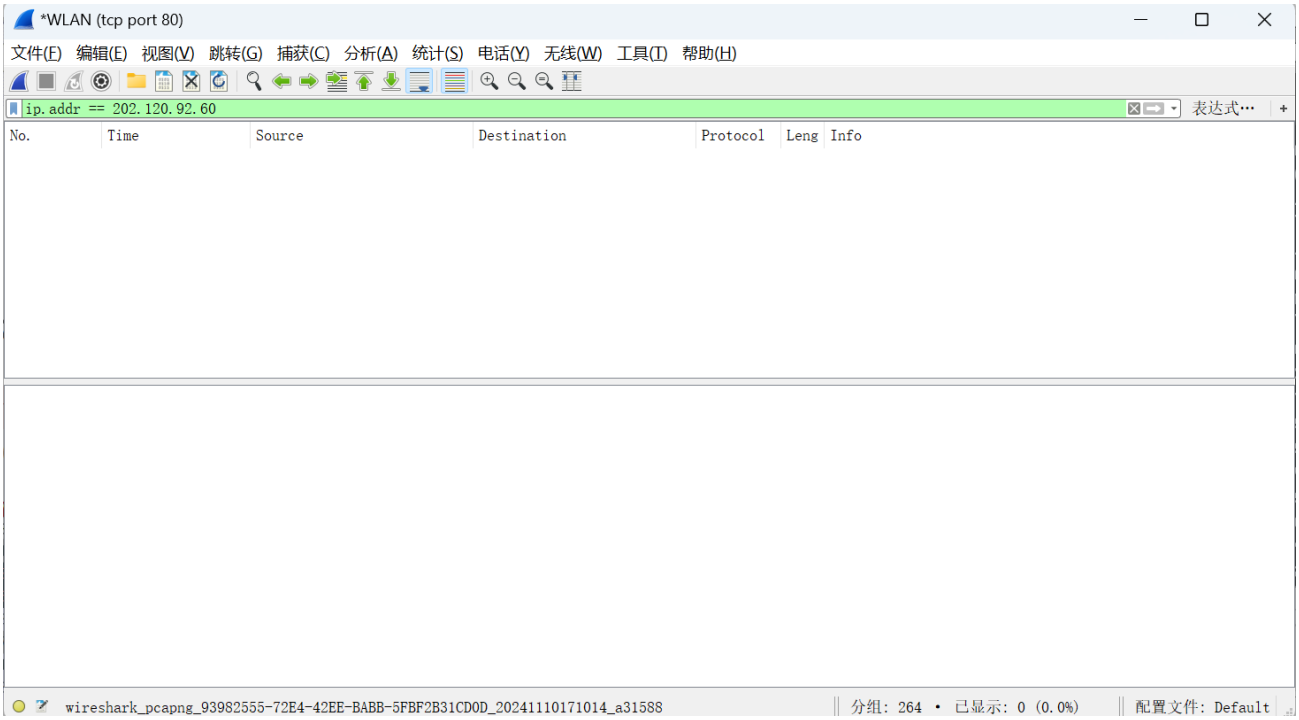
首先查看目标地址的 ip，这样可以利用 wireshark 内置过滤器（wireshark 具有过滤器的功能，这样就没必要在抓包前关闭其他网页了）有效过滤不想要的包：

```
PS C:\> nslookup www.ecnu.edu.cn
服务器:  moon.ecnu.edu.cn
Address:  202.120.80.2

非权威应答:
名称:    www.ecnu.edu.cn
Addresses:  2001:da8:8005:a492::60
          202.120.92.60
```

可以看到学校服务器的 ipv4 地址是 202.120.92.60.

接着进入 wireshark，设置好监听接口 wlan（连的是学校网络）和端口 80，并设置好过滤器条件为学校服务器 ip：



接着点击左上角的蓝色小鲨鱼，开始捕获目标数据包，并在命令行输入 wget http://www.ecnu.edu.cn，并回车：

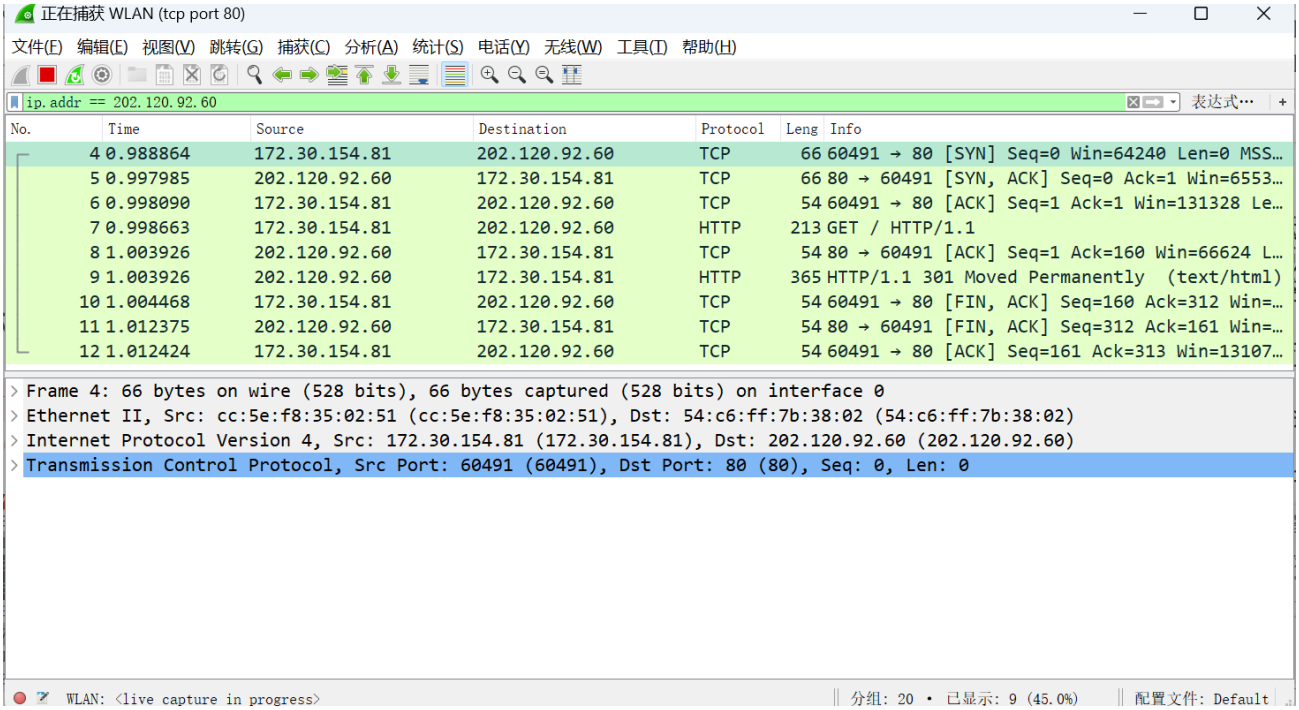
```
PS C:\> wget http://www.ecnu.edu.cn


StatusCode        : 200
StatusDescription : OK
Content           : ï»¿<!DOCTYPE html>
                    <html>
                    <head>
                    <meta charset="UTF-8">
                    <meta name="viewport" content="width=device-width, initia
                    l-scale=1.0">

                    <meta name="description" content="" />
                    <title>åäˌåˌèå¤§å-¦<...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    X-Frame-Options: SAMEORIGIN
                    X-XSS-Protection: 1; mode=block
                    X-Content-Type-Options: nosniff
                    Referer-Policy: no-referer-when-downgrade
                    X-Download-Options: n...
Forms             : {au3a}
Headers           : {[Connection, keep-alive], [X-Frame-Options, SAMEORIGIN],
                     [X-XSS-Protection, 1; mode=block], [X-Content-Type-Optio
                    ns, nosniff]...}
Images            : {@{innerHTML=; innerText=; outerHTML=<IMG alt="" src="ima
                    ges/fx300.jpg">; outerText=; tagName=IMG; alt=; src=image
                    s/fx300.jpg}, @{innerHTML=; innerText=; outerHTML=<IMG al
                    t=åäˌåˌèå¤§å&shy;¦ src="images/logo.svg" width=51
                    0 height=90>; outerText=; tagName=IMG; alt=åäˌåˌèå
                    ¤§å&shy;¦; src=images/logo.svg; width=510; height=90},
                    @{innerHTML=; innerText=; outerHTML=<IMG src="images/5992
                    59db264d28d87e3de4774f23f63.png">; outerText=; tagName=IM
                    G; src=images/599259db264d28d87e3de4774f23f63.png}, @{inn
                    erHTML=; innerText=; outerHTML=<IMG alt="" src="/__local/
                    A/08/49/E29C293ABB78CA6CB167D883813_096E4AA7_2C8A3.jpg" w
                    idth=820 height=420>; outerText=; tagName=IMG; alt=; src=
                    /__local/A/08/49/E29C293ABB78CA6CB167D883813_096E4AA7_2C8
                    A3.jpg; width=820; height=420}...}
InputFields       : {@{innerHTML=; innerText=; outerHTML=<INPUT id=lucenenews
                    searchkey1086 type=hidden name=lucenenewssearchkey>; oute
                    rText=; tagName=INPUT; id=lucenenewssearchkey1086; type=h
                    idden; name=lucenenewssearchkey}, @{innerHTML=; innerText
                    =; outerHTML=<INPUT id=_lucenesearchtype1086 type=hidden
                    value=1 name=_lucenesearchtype>; outerText=; tagName=INPU
                    T; id=_lucenesearchtype1086; type=hidden; value=1; name=_
                    lucenesearchtype}, @{innerHTML=; innerText=; outerHTML=<I
                    NPUT id=searchScope1086 type=hidden value=1 name=searchSc
                    ope>; outerText=; tagName=INPUT; id=searchScope1086; type
                    =hidden; value=1; name=searchScope}, @{innerHTML=; innerT
                    ext=; outerHTML=<INPUT id=showkeycode1086 class=search-tx
                    t name=showkeycode placeholder="è¯·è¾¥å¥å³é®è¯æ£ç´¢
```

```
                    ç¸å³åå®¹">; outerText=; tagName=INPUT; id=showkeycode
                    1086; class=search-txt; name=showkeycode; placeholder=è¯·è¾å¥å³é®è¯æ£ç´¢ç¸å³åå®¹}...}
Links             : {@{innerHTML=English; innerText=English; outerHTML=<A onc
                    lick='_addDynClicks("wburl", 1726036883, 1021)' title=""
                    href="http://english.ecnu.edu.cn/" target=_blank>English<
                    /A>; outerText=English; tagName=A; onclick=_addDynClicks(
                    "wburl", 1726036883, 1021); title=; href=http://english.e
                    cnu.edu.cn/; target=_blank}, @{innerHTML=FranÃ§ais; inne
                    rText=FranÃ§ais; outerHTML=<A onclick='_addDynClicks("wb
                    url", 1726036883, 1473)' title="" href="http://french.ecn
                    u.edu.cn/" target=_blank>FranÃ§ais</A>; outerText=FranÃ
                    §ais; tagName=A; onclick=_addDynClicks("wburl", 17260368
                    83, 1473); title=; href=http://french.ecnu.edu.cn/; targe
                    t=_blank}, @{innerHTML=Ð ÑÑÐ°Ð¸Ð¹; innerText=Ð Ñ
                    ÑÐ°Ð¸Ð¹; outerHTML=<A title="" href="https://russian.
                    ecnu.edu.cn" target=_blank>Ð ÑÑÐ°Ð¸Ð¹</A>; outer
                    Text=Ð ÑÑÐ°Ð¸Ð¹; tagName=A; title=; href=https://russ
                    ian.ecnu.edu.cn; target=_blank}, @{innerHTML=å&shy;¦æ&nbs
                    p;¡æ¦åµ; innerText=å-¦æ ¡æ¦åµ; outerHTML=<A href="wzc
                    d/xxgk.htm" target=_blank>å&shy;¦æ ¡æ¦åµ</A>; oute
                    rText=å-¦æ ¡æ¦åµ; tagName=A; href=wzcd/xxgk.htm; target
                    =_blank}...}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 138738
```

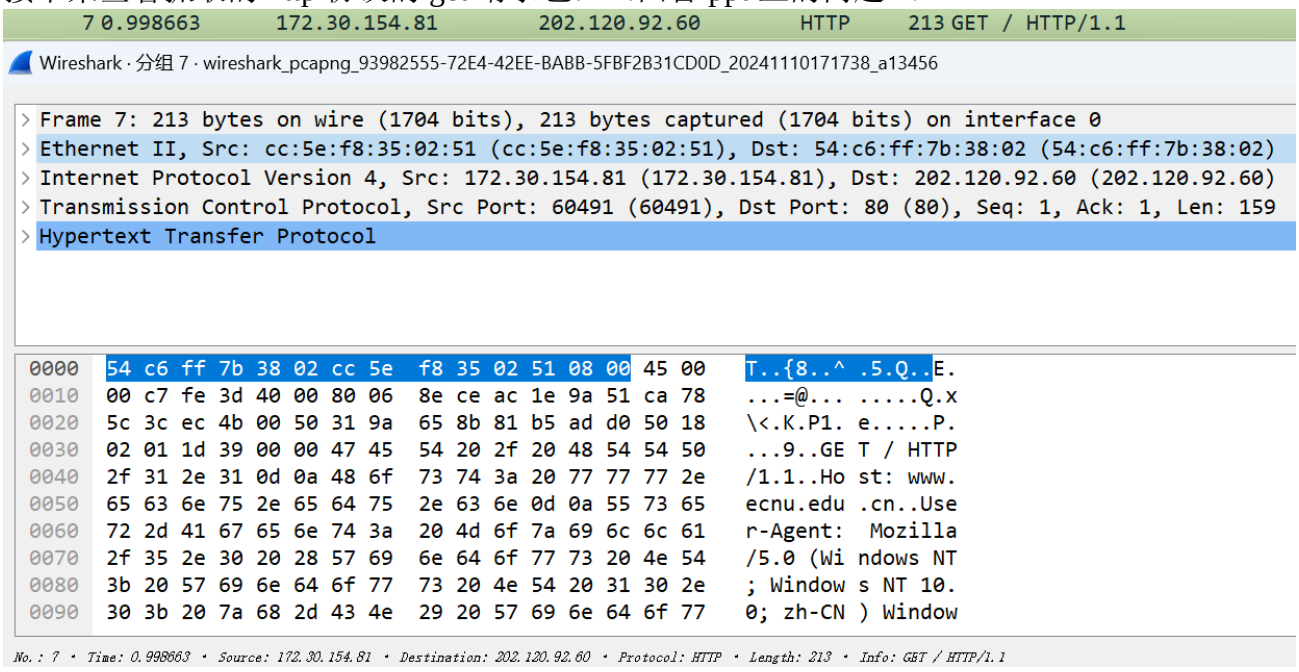
可以发现 statuscode 是 200，说明我们 get 请求成功，接下来查看 wireshark 内容，得到如下结果：



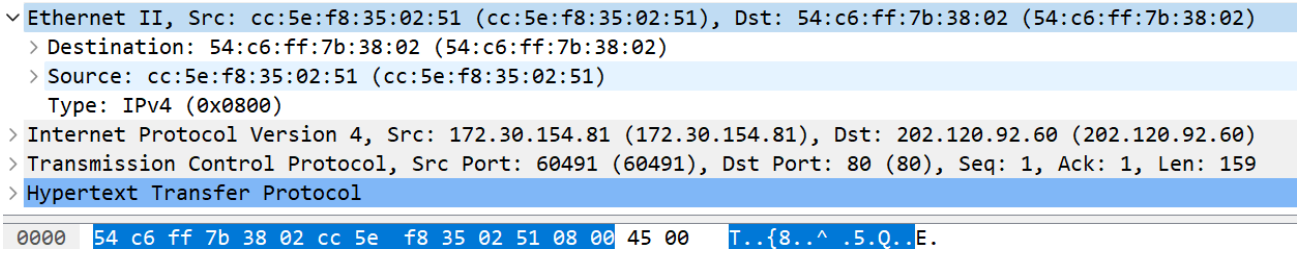
可以看到前三个 tcp 包作为三次握手的过程，最后三个包是四次挥手的过程（此处并不是课本上的四个 tcp 包，原因在于**开启了延时 ack 机制 ，导致收到第一个 fin 之后（上图的第十个封包，由本机发出），发送 ack 的条件不能满足立即发送 ack 的条件，导致 ack 的发送被延时了，在延时的过程中，应用如果确认没数据要发，并且也要关闭此连接的情况下，会触发发送 fin，这个 fin 就会和之前的 ack 合并被发出，也就是说，第二次和第三次挥手**

（服务器向客户发送的）合并了。资料参考：《Linux 的 TCP 实现之：四次挥手》以及 stackoverflow 的一个帖子：https://stackoverflow.com/questions/15182106/what-is-the-reason-and-how-to-avoid-the-fin-ack-rst-and-rst-ack）

接下来查看抓取的 http 协议的 get 请求包：（回答 ppt 上的问题 1）



对于以太网头部（数据链路层）：



可以看到其占有 14 个字节（16 进制高亮部分），
由目标地址和发送地址以及下一层协议的标识码（6+6+2）组成

对于 ip 头部（网络层）：

```
Internet Protocol Version 4, Src: 172.30.154.81 (172.30.154.81), Dst: 202.120.92.60 (202.120.92.60)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 199
    Identification: 0xfe3d (65085)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  > Header checksum: 0x8ece [validation disabled]
    Source: 172.30.154.81 (172.30.154.81)
    Destination: 202.120.92.60 (202.120.92.60)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> Transmission Control Protocol, Src Port: 60491 (60491), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 159
> Hypertext Transfer Protocol
```

```
0000  54 c6 ff 7b 38 02 cc 5e  f8 35 02 51 08 00 45 00   T..{8..^ .5.Q..E.
0010  00 c7 fe 3d 40 00 80 06  8e ce ac 1e 9a 51 ca 78   ...=@... .....Q.x
0020  5c 3c ec 4b 00 50 31 9a  65 8b 81 b5 ad d0 50 18   \<.K.P1. e.....P.
```

Ip 头部记录了更多的信息，包括 ip 封包后的总长度，源 ip 地址和目的 ip 地址，当然还有传输层协议的标识码（06：tcp 协议）以及刚刚学的纠错码 checksum 等，总共 20 字节。

对于 tcp 头部（传输层）：

```
Transmission Control Protocol, Src Port: 60491 (60491), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 159
    Source Port: 60491
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 159]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 160    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  > Flags: 0x018 (PSH, ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
  > Checksum: 0x1d39 [validation disabled]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
```

```
0000  54 c6 ff 7b 38 02 cc 5e  f8 35 02 51 08 00 45 00   T..{8..^ .5.Q..E.
0010  00 c7 fe 3d 40 00 80 06  8e ce ac 1e 9a 51 ca 78   ...=@... .....Q.x
0020  5c 3c ec 4b 00 50 31 9a  65 8b 81 b5 ad d0 50 18   \<.K.P1. e.....P.
0030  02 01 1d 39 00 00 47 45  54 20 2f 20 48 54 54 50   ...9..GE T / HTTP
```

传输的字节数和虽然和 ip 头部一样（都是 20 字节），但是 tcp 的源地址和目标地址精确到了相应主机上的进程（即端口 port），同时也有 checksum 纠错码，以及各种 flag 来确定传输的作用（是建立连接还是传数据还是断开连接，例如 SYN 是建立连接的请求）

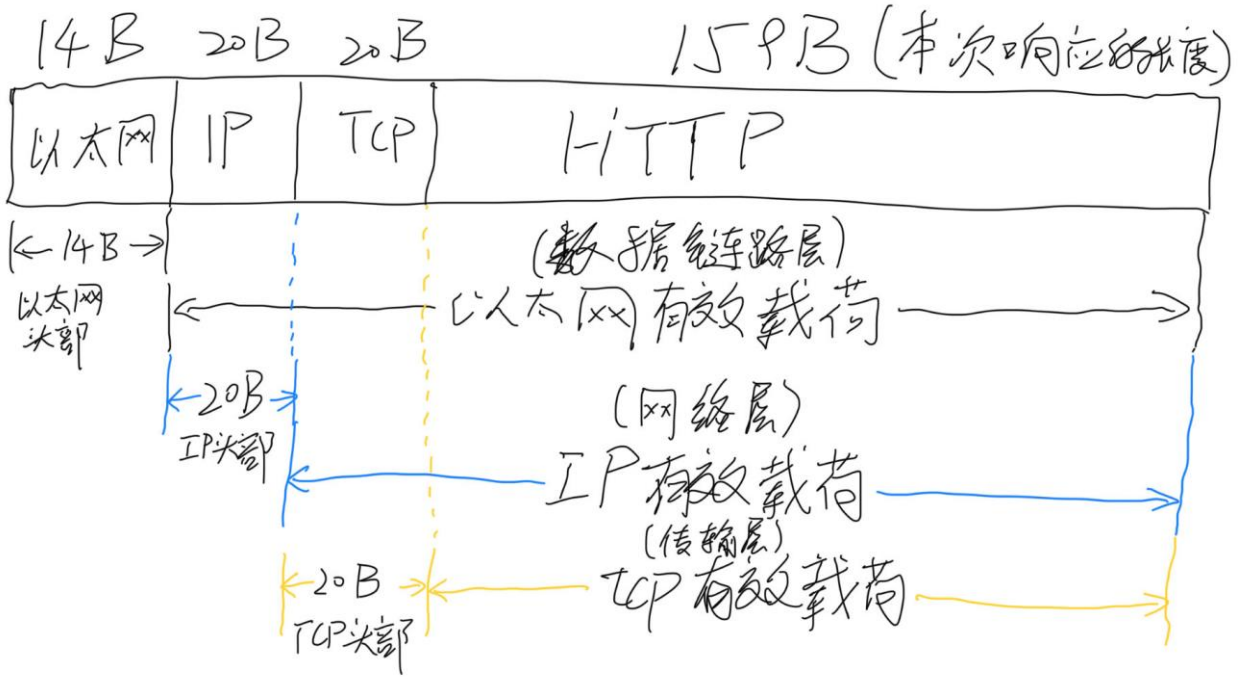
接下来就是应用层的 http 协议标头以及具体传输的内容：

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: www.ecnu.edu.cn\r\n
  User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; zh-CN) WindowsPowerShell/5.1.22621
  Accept-Encoding: gzip\r\n
  \r\n
  [Full request URI: http://www.ecnu.edu.cn/]
  [HTTP request 1/1]
  [Response in frame: 9]
```

```
0010  00 c7 fe 3d 40 00 80 06  8e ce ac 1e 9a 51 ca 78   ...=@... .....Q.x
0020  5c 3c ec 4b 00 50 31 9a  65 8b 81 b5 ad d0 50 18   \<.K.P1. e.....P.
0030  02 01 1d 39 00 00 47 45  54 20 2f 20 48 54 54 50   ...9..GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 77 77 77 2e   /1.1..Ho st: www.
0050  65 63 6e 75 2e 65 64 75  2e 63 6e 0d 0a 55 73 65   ecnu.edu .cn..Use
0060  72 2d 41 67 65 6e 74 3a  20 4d 6f 7a 69 6c 6c 61   r-Agent:  Mozilla
0070  2f 35 2e 30 20 28 57 69  6e 64 6f 77 73 20 4e 54   /5.0 (Wi ndows NT
0080  3b 20 57 69 6e 64 6f 77  73 20 4e 54 20 31 30 2e   ; Window s NT 10.
0090  30 3b 20 7a 68 2d 43 4e  29 20 57 69 6e 64 6f 77   0; zh-CN ) Window
00a0  73 50 6f 77 65 72 53 68  65 6c 6c 2f 35 2e 31 2e   sPowerSh ell/5.1.
00b0  32 32 36 32 31 2e 33 36  37 32 0d 0a 41 63 63 65   22621.36 72..Acce
00c0  70 74 2d 45 6e 63 6f 64  69 6e 67 3a 20 67 7a 69   pt-Encod ing: gzi
00d0  70 0d 0a 0d 0a                                    p....
```

可以看到发出 get 请求的是一个浏览器（wget 是伪装成浏览器发送请求的）

画出协议层嵌套结构的图如下（回答 ppt 上的问题二）：



分析协议的开销（回答 ppt 上的问题三以及问题四的第一小问）：

认为 http 协议包含的信息都是有效的数据，以太网，ip，tcp 的头部视作开销，计算有效数据占比，由于 tcp 协议有三次握手的过程，需要将后两次握手的开销和我们确认收到的 tcp 包的开销算上（尽管这三次没有有效数据传输，同时注意我们发起请求 tcp 连接的最初始的那个包不算，因为此包是我们发送给服务器请求连接的，后面发回来才说明允许我们连

接）：

```
5 0.997985    202.120.92.60    172.30.154.81    TCP     66 80 → 60491 [SYN, ACK] Seq=0 Ack=1 Win=6553…
6 0.998090    172.30.154.81    202.120.92.60    TCP     54 60491 → 80 [ACK] Seq=1 Ack=1 Win=131328 Le…
7 0.998663    172.30.154.81    202.120.92.60    HTTP   213 GET / HTTP/1.1
8 1.003926    202.120.92.60    172.30.154.81    TCP     54 80 → 60491 [ACK] Seq=1 Ack=160 Win=66624 L…
```

由之前的协议分析可知，有效的数据字节数为 159 字节，而总字节数为 66+54+213+54=387 个字节，利用率为：159/387*100%=41.085%，这只是本次实验的包的占比，实际包的占比要比这个高很多，http 部分理论最高可达到 1500 字节，这样的话理论最高利用率为 1500/(387-159+1500)*100%=86.806%

五、实验结果总结

ppt 问题 4 第二小问（对于下载的主要部分中的每一个包，我们需要分析 Ethernet，IP 和 TCP 的开销，和有用的 HTTP 数据的开销，你认为这种开销是必要的吗？）：

是有必要的，tcp 头部的 flag，就是用来确认连接，确认是否接收到数据，断开连接的，tcp 也有序列码和 checksum 等机制来保证传输，这些都是 tcp 作为可靠的面向连接的必要条件，因此是必要的。

ppt 问题 5（以太网头部中哪一部分是解复用（解复用： 找到正确的上一层协议来处理到达的包的行为叫做 解复用）键并且告知它的下一个高层指的是 IP，在这一包内哪一个值可以表示 IP？）：

在之前分析以太网头部的时候，有下图：

```
Ethernet II, Src: cc:5e:f8:35:02:51 (cc:5e:f8:35:02:51), Dst: 54:c6:ff:7b:38:02 (54:c6:ff:7b:38:02)
  Destination: 54:c6:ff:7b:38:02 (54:c6:ff:7b:38:02)
  Source: cc:5e:f8:35:02:51 (cc:5e:f8:35:02:51)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 172.30.154.81 (172.30.154.81), Dst: 202.120.92.60 (202.120.92.60)
Transmission Control Protocol, Src Port: 60491 (60491), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 159
Hypertext Transfer Protocol

0000  54 c6 ff 7b 38 02 cc 5e  f8 35 02 51 08 00 45 00   T..{8..^ .5.Q..E.
```

可以看到 Ethernet II 下面有一个 type，就是 ipv4（即 0x0800），这个部分是解复用，表示 ip。

ppt 问题 6（IP 头部中哪一部分是解复用键并且告知它的下一个高层指的是 TCP，在这一包内哪一个值可以表示 TCP？）

之前分析的 ip 头部为：

```
Internet Protocol Version 4, Src: 172.30.154.81 (172.30.154.81), Dst: 202.120.92.60 (202.120.92.60)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 199
  Identification: 0xfe3d (65085)
  Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 128
  Protocol: TCP (6)
  Header checksum: 0x8ece [validation disabled]
  Source: 172.30.154.81 (172.30.154.81)
  Destination: 202.120.92.60 (202.120.92.60)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 60491 (60491), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 159
Hypertext Transfer Protocol

0010  00 c7 fe 3d 40 00 80 06  8e ce ac 1e 9a 51 ca 78   ...=@... .....Q.x
0020  5c 3c ec 4b 00 50 31 9a  65 8b 81 b5 ad d0 50 18   \<.K.P1. e.....P.
```

可以看到列表中有一个 Protocol:TCP(6)，表示 tcp，为解复用键

ppt 问题 7（查看不包含高层数据的短 TCP 数据包，查看它发往哪？不携带高层数据的数据包有用吗？）

不包含数据的 tcp 包一般是用于确认链接和断开连接的包（三次握手和四次挥手），并且一般是在客户机和服务器之间发送

ppt 问题 8（在经典的分层模型中，低层字段包装到高层数据包外面，成为一条新消息。但这并非总是如此,Web 响应（一个包含 HTTP 标头和 HTTP 有效负载的 HTTP 消息）可能被转换为多个较低层的消息（即多个 TCP 数据包）。假设你为 Web 响应的第一个和最后一个 TCP 数据包绘制了数据包结构，那么该结构与经典分层模型有什么不同？）

解答该问题需要了解 tcp 头部的结构，tcp 头部结构如下：

第一个 tcp 实际上没有数据要传，也就是说，tcp 包内除了其他协议的头部就是 tcp 包的头部了，同时第一个 tcp 是客户向服务器请求连接的，请求连接时要将 flag 中的 SYN 位置一，其他 flag 位置零

对于最后一个 tcp 包，和第一个一样，要将 flag 中的相应位搞对（四次挥手分别是：

第一次：ACK 和 FIN 位为一

第二次：ACK 为一

第三次：ACK 和 FIN 位为一

第四次：ACK 为一。）

除此之外，还要注意序号和确认号的变化（二者均是相对的）

ppt 问题 9：在上述经典分层模型中，低层字段包装到高层数据包外面，如果较低层添加加密，此模型将如何更改？

加密的方式在数据传输前确定即可（例如 ssh，服务器上要有公匙，本地有私匙才能传数据，实例：github 上 push 代码）

ppt 问题 10：在上述经典分层模型中，低层字段包装到高层数据包外面,如果较低的层添加压缩，此模型将如何更改？

将压缩协议包含在协议的头部，这样能传输更多的数据，同时在 header 中也能够得知压缩方式进行解压缩。

六、个人总结

本次实验很好地让我理解了 tcp 协议的三次握手和四次挥手，同时详细了解了 tcp 头部的实现细节，对于 http 协议的 get 请求的协议栈有了更深的理解，除此之外，本人还尝试使用 python 爬虫代替 wget 模拟上网（加了一些克制反爬虫的机制），但根据 fiddle 的抓包情况来看（wireshark 根本没抓到），没有成功，我们学校网站有反爬虫的机制（那 wget 是怎么绕过的？）。总的来说，本次实验我收获较多，对各个层次之间了解更为深刻。