计算机网络

课程设计 2——网络抓包分析

TCP 报文分析

学校：西南交通大学

指导老师：谭献海

姓名：谭梓琦

学院：信息科学与技术学院

班级：物联网工程 1 班

学号：2015112210

目录

1TCP 协议简介

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议，由 IETF 的 RFC 793 定义。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能，用户数据报协议（UDP）是同一层内另一个重要的传输协议。

1.1 实行标准

TCP/IP(Transmission Control Protocol/Internet Protocol) 即传输控制协议/网间协议，是一个工业标准的协议集，它是为广域网（WAN）设计的。它是由 ARPANET 网的研究机构发展起来的。

TCP/IP 的标准在一系列称为 RFC 的文档中公布。文档由技术专家、特别工作组、或 RFC 编辑修订。公布一个文档时，该文档被赋予一个 RFC 编号，如 RFC959（FTP 的说明文档）、RFC793（TCP 的说明文档）、RFC791（IP 的说明文档）等。最初的 RFC 一直保留而从来不会被更新，[1] 如果修改了该文档，则该文档又以一个新号码公布。因此，重要的是要确认你拥有了关于某个专题的最新 RFC 文档。通常在 RFC 的开头部分，有相关 RFC 的更新(update)、排错（errata）、作废（obsolete）信息，提示读者信息的时效性。

1.2 功能

当应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，TCP 则把数据流分割成适当长度的报文段，最大传输段大小（MSS）通常受该计算机连接的网络的数据链路层的最大传送单元（MTU）限制。之后 TCP 把数据包传给 IP 层，由它来通过网络将包传送给接收端实体的 TCP 层。

TCP 为了保证报文传输的可靠，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的字节发回一个相应的确认(ACK)；如果发送端实体在合理的往返时延(RTT)内未收到确认，那么对应的数据（假设丢失了）将会被重传。

在数据正确性与合法性上，TCP 用一个校验和函数来检验数据是否有错误，在发送和接收时都要计算校验和；同时可以使用 md5 认证对数据进行加密。

在保证可靠性上，采用超时重传和捎带确认机制。

在流量控制上，采用滑动窗口协议，协议中规定，对于窗口内未经确认的分组需要重传。

在拥塞控制上，采用广受好评的 TCP 拥塞控制算法（也称 AIMD 算法）。该算法主要包括三个主要部分：1）加性增、乘性减；2）慢启动；3）对超时事件做出反应。

1.3 可靠性

TCP 提供一种面向连接的、可靠的字节流服务。面向连接意味着两个使用 TCP 的应用（通常是一个客户和一个服务器）在彼此交换数据包之前必须先建立一个 TCP 连接。这一过程与打电话很相似，先拨号振铃，等待对方摘机说“喂”，然后才说明是谁。在一个 TCP 连接中，仅有两方进行彼此通信。广播和多播不能用于 TCP。

TCP 通过下列方式来提供可靠性：

1．应用数据被分割成 TCP 认为最适合发送的数据块。这和 UDP 完全不同，应用程序产生的数据长度将保持不变。由 TCP 传递给 IP 的信息单位称为报文段或段（segment）。

2．当 TCP 发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。当 TCP 收到发自 TCP 连接另一端的数据，它将发送一个确认。TCP 有延迟确认的功能，在此功能没有打开，则是立即确认。功能打开，则由定时器触发确认时间点。

3．TCP 将保持它首部和数据的检验和。这是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果收到段的检验和有差错，TCP 将丢弃这个报文段和不确认收到此报文段（希望发端超时并重发）。

4．既然 TCP 报文段作为 IP 数据报来传输，而 IP 数据报的到达可能会失序，因此 TCP 报文段的到达也可能会失序。如果必要，TCP 将对收到的数据进行重新排序，将收到的数据以正确的顺序交给应用层。

5．既然 IP 数据报会发生重复，TCP 的接收端必须丢弃重复的数据。

6．TCP 还能提供流量控制。TCP 连接的每一方都有固定大小的缓冲空间。TCP 的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。

两个应用程序通过 TCP 连接交换 8bit 字节构成的字节流。TCP 不在字节流中插入记录标识符。我们将这称为字节流服务（bytestreamservice）。如果一方的应用程序先传 10 字节，又传 20 字节，再传 50 字节，连接的另一方将无法了解发方每次发送了多少字节。只要自己的接收缓存没有塞满，TCP 接收方将有多少就收多少。一端将字节流放到 TCP 连接上，同样的字节流将出现在 TCP 连接的另一端。

另外，TCP 对字节流的内容不作任何解释。TCP 不知道传输的数据字节流是二进制数据，还是 ASCⅡ字符、EBCDIC 字符或者其他类型数据。对字节流的解释由 TCP 连接双方的应用层解释。

这种对字节流的处理方式与 Unix 操作系统对文件的处理方式很相似。Unix 的内核对一个应用读或写的内容不作任何解释，而是交给应用程序处理。对 Unix 的内核来说，它无法区分一个二进制文件与一个文本文件。

1.3.1 重传策略

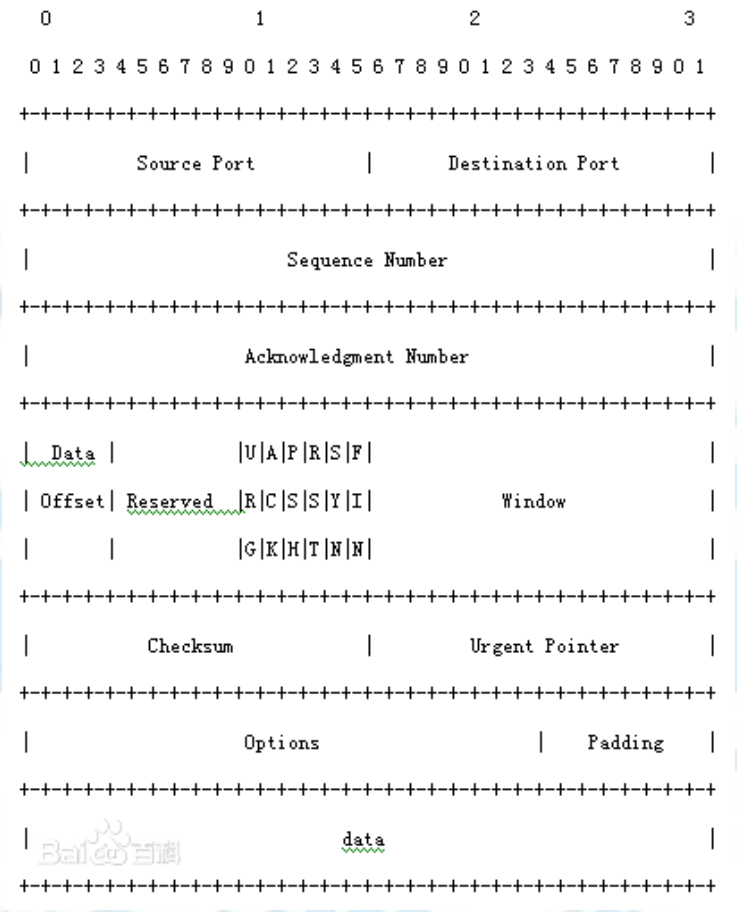
TCP 协议用于控制数据段是否需要重传的依据是设立重发定时器。在发送一个数据段的同时启动一个重传，如果在重传超时前收到确认(Acknowlegement)就关闭该重传，如果重传超时前没有收到确认，则重传该数据段。在选择重发时间的过程中，TCP 必须具有自适应性。它需要根据互联网当时的通信情况，给出合适的重发时间。

这种重传策略的关键是对定时器初值的设定。采用较多的算法是 Jacobson 于 1988 年提出的一种不断调整超时时间间隔的动态算法。其工作原理是：对每条连接 TCP 都保持一个变量 RTT（Round Trip Time），用于存放当前到目的端往返所需要时间最接近的估计值。当发送一个数据段时，同时启动连接的定时器，如果在定时器超时前确认到达，则记录所需要的时间（M），并修正 RTT 的值，如果定时器超时前没有收到确认，则将 RTT 的值增加 1 倍。通过测量一系列的 RTT（往返时间）值，TCP 协议可以估算数据包重发前需要等待的时间。在估计该连接所需的当前延迟时通常利用一些统计学的原理和算法（如 Karn 算法），从而得到 TCP 重发之前需要等待的时间值。

1.3.2 窗口确认

TCP 的一项功能就是确保每个数据段都能到达目的地。位于目的主机的 TCP 服务对接受到的数据进行确认，并向源应用程序发送确认信息。使用数据报头序列号以及确认号来确认已收到包含在数据段的相关的数据字节。TCP 在发回源设备的数据段中使用确认号，指示接收设备期待接收的下一字节。这个过程称为期待确认。源主机在收到确认消息之前可以传输的数据的大小称为窗口大小。用于管理丢失数据和流量控制。这些变化如右图所示。

2TCP 报文封装



TCP 报文格式

16 位源端口号：包含初始化通信的端口。源端口和源 IP 地址的作用是标识报文的返回地址。

16 位目的端口号：定义传输的目的。这个端口指明报文接收计算机上的应用程序地址接口。

32 位序号：由接收端计算机使用，重新分段的报文成最初形式。当 SYN 出现，序列码实际上是初始序列码（Initial Sequence Number，ISN），而第一个数据字节是 ISN+1。这个序列号（序列码）可用来补偿传输中的不一致。

32 位确认序号：由接收端计算机使用，重组分段的报文成最初形式。如果设置了 ACK 控制位，这个值表示一个准备接收的包的序列码。

4 位首部长度：包括 TCP 头大小，指示何处数据开始。

保留（6 位）：这些位必须是 0。为了将来定义新的用途而保留。

6 位标志域。表示为：紧急标志、有意义的应答标志、推、重置连接标志、同步序列号标志、完成发送数据标志。按照顺序排列是：URG、ACK、PSH、RST、SYN、FIN。

16 位窗口大小：用来表示想收到的每个 TCP 数据段的大小。TCP 的流量控制由连接的每一端通过声明的窗口大小来提供。窗口大小为字节数，起始于确认序号字段指明的值，这个值是接收端正期望接收的字节。窗口大小是一个 16 字节字段，因而窗口大小最大为 65535 字节。

16 位校验和：16 位 TCP 头。源机器基于数据内容计算一个数值，收信息机要与源机器数值 结果完全一样，从而证明数据的有效性。检验和覆盖了整个的 TCP 报文段：这是一个强制性的字段，一定是由发送端计算和存储，并由接收端进行验证的。

16 位紧急指针：指向后面是优先数据的字节，在 URG 标志设置了时才有效。如果 URG 标志没有被设置，紧急域作为填充。加快处理标示为紧急的数据段。

选项：长度不定，但长度必须为 1 个字节。如果没有选项就表示这个 1 字节的域等于 0。

数据：该 TCP 协议包负载的数据。

在上述字段中，6 位标志域的各个选项功能如下。

URG：紧急标志。紧急标志为"1"表明该位有效。

ACK：确认标志。表明确认编号栏有效。大多数情况下该标志位是置位的。TCP 报头内的确认编号栏内包含的确认编号（w+1）为下一个预期的序列编号，同时提示远端系统已经成功接收所有数据。

PSH：推标志。该标志置位时，接收端不将该数据进行队列处理，而是尽可能快地将数据转由应用处理。在处理 Telnet 或 rlogin 等交互模式的连接时，该标志总是置位的。

RST：复位标志。用于复位相应的 TCP 连接。

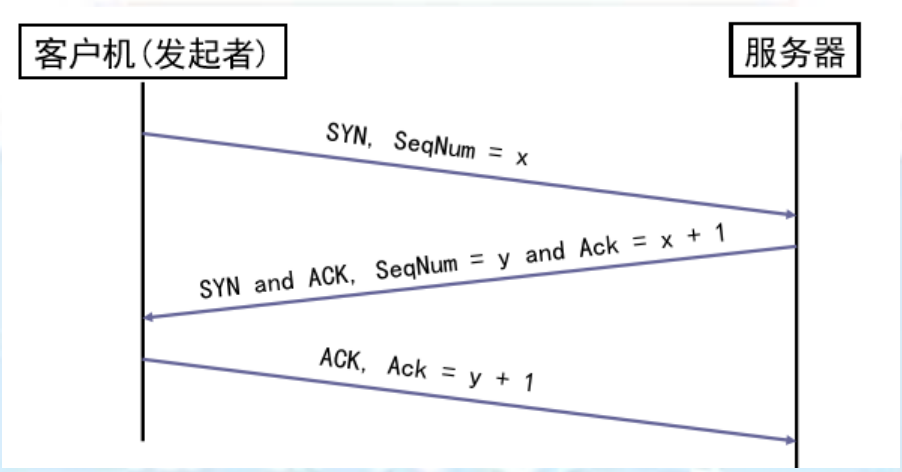
SYN：同步标志。表明同步序列编号栏有效。该标志仅在三次握手建立 TCP 连接时有效。它提示 TCP 连接的服务端检查序列编号，该序列编号为 TCP 连接初始端（一般是客户端）的初始序列编号。在这里，可以把 TCP 序列编号看作是一个范围从 0 到 4，294，967，295 的 32 位计数器。通过 TCP 连接交换的数据中每一个字节都经过序列编号。在 TCP 报头中的序列编号栏包括了 TCP 分段中第一个字节的序列编号。

FIN：结束标志。

3TCP 工作流程

3.1TCP 三次握手

三次握手（Three-Way Handshake）即建立 TCP 连接，就是指建立一个 TCP 连接时，需要客户端和服务端总共发送 3 个包以确认连接的建立。



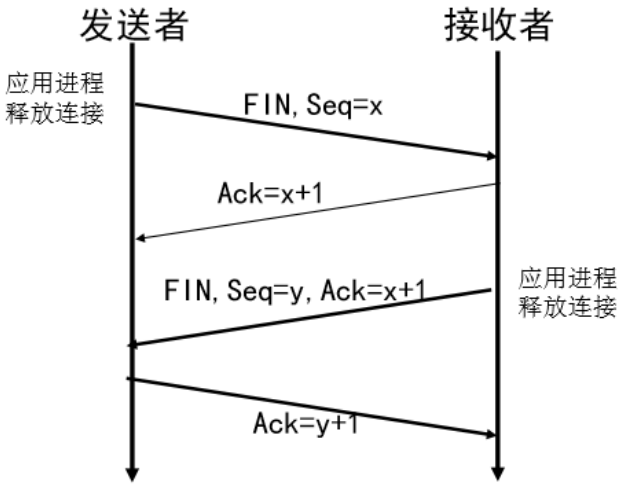
TCP 三次握手

第一次握手：Client 将标志位 SYN 置为 1，随机产生一个值 seq = x，并将该数据包发送给 Server，Client 进入 SYN_SENT 状态，等待 Server 确认。

第二次握手：Server 收到数据包后由标志位 SYN = 1 知道 Client 请求建立连接，Server 将标志位 SYN 和 ACK 都置为 1，Ack = x + 1，随机产生一个值 seq = y，并将该数据包发送给 Client 以确认连接请求，Server 进入 SYN_RCVD 状态。

第三次握手：Client 收到确认后，检查 Ack 是否为 x + 1，ACK 是否为 1，如果正确则将标志位 ACK 置为 1，Ack = y + 1，并将该数据包发送给 Server，Server 检查 Ack 是否为 y+1，ACK 是否为 1，如果正确则连接建立成功，Client 和 Server 进入 ESTABLISHED 状态，完成三次握手，随后 Client 与 Server 之间可以开始传输数据了。

3.2TCP 四次挥手



TCP 四次挥手

四次挥手（Four-Way Wavehand）即终止 TCP 连接，就是指断开一个 TCP 连接时，需要客户端和服务端总共发送 4 个包以确认连接的断开。

注意：连接释放的 4 个数据包并不是连续的，是否发送也与服务器和客户机的软件设计有关。

第一次挥手：Client 发送一个 FIN，用来关闭 Client 到 Server 的数据传送，Client 进入 FIN_WAIT_1 状态。

第二次挥手：Server 收到 FIN 后，发送一个 ACK 给 Client，确认序号为收到序号+1（与 SYN 相同，一个 FIN 占用一个序号），Server 进入 CLOSE_WAIT 状态。

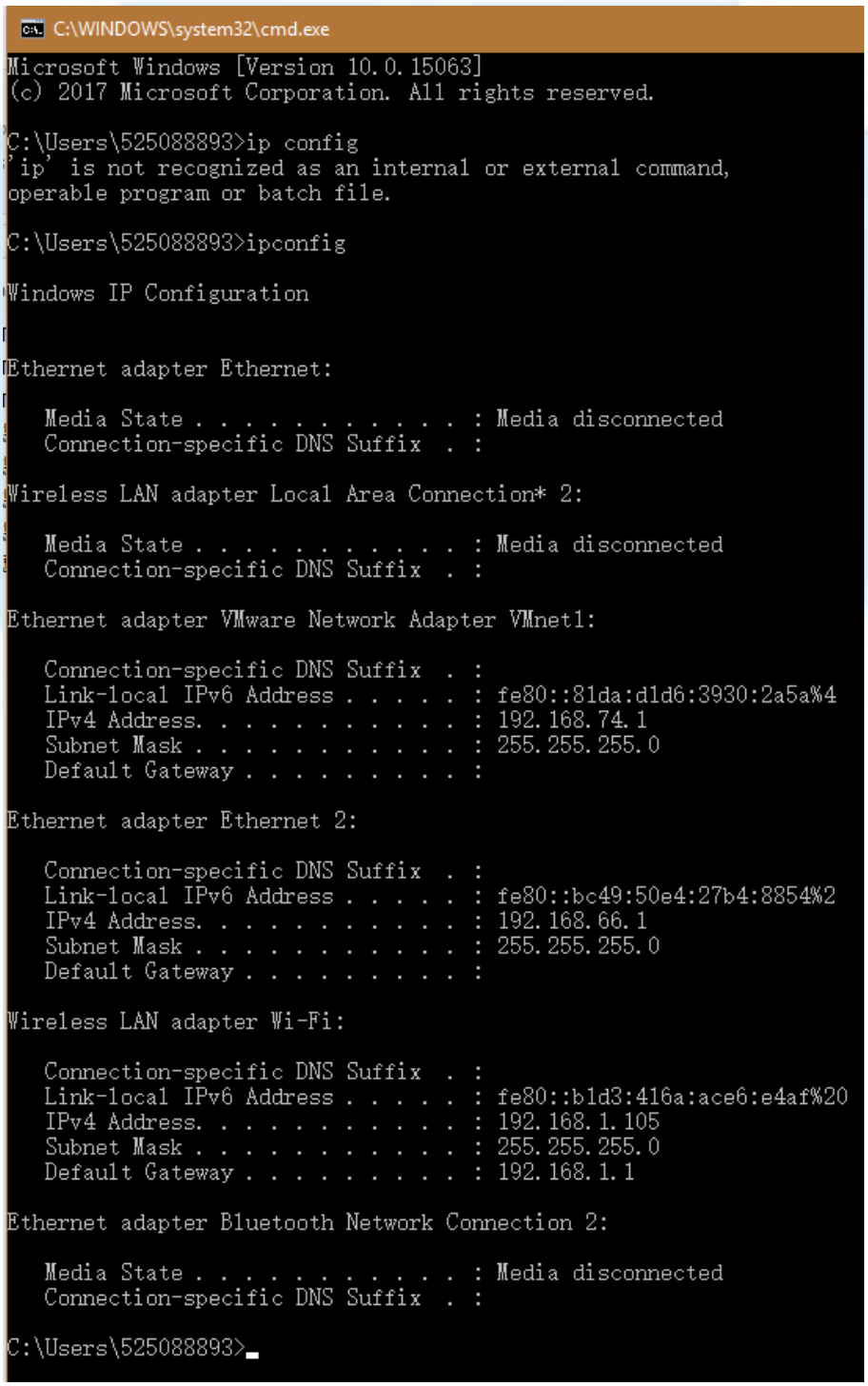
第三次挥手：Server 发送一个 FIN，用来关闭 Server 到 Client 的数据传送，Server 进入 LAST_ACK 状态。

第四次挥手：Client 收到 FIN 后，Client 进入 TIME_WAIT 状态，接着发送一个 ACK 给 Server，确认序号为收到序号+1，Server 进入 CLOSED 状态，完成四次挥手。

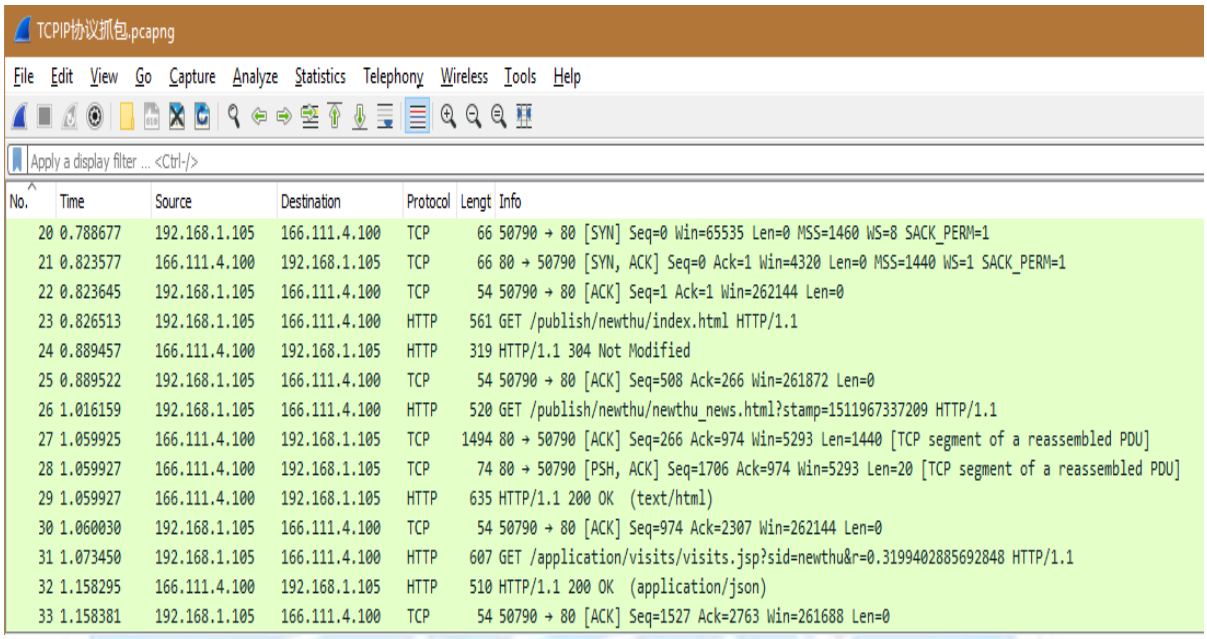
4TCP 报文抓包实例分析——访问清华大学官方网站

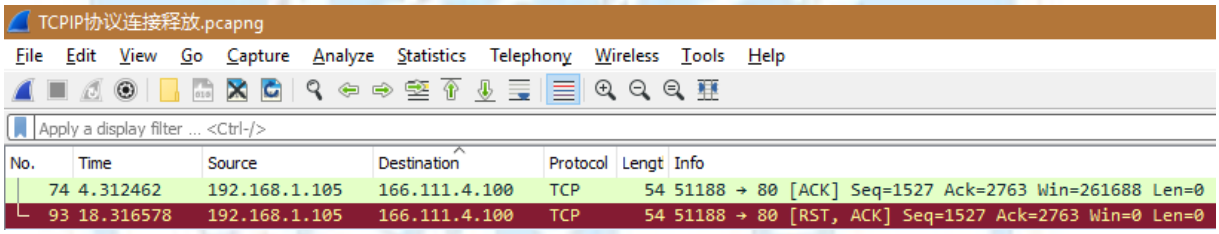
访问 http://www.tsinghua.edu.cn/publish/newthu/index.html

4.1 本地计算机 IP 配置

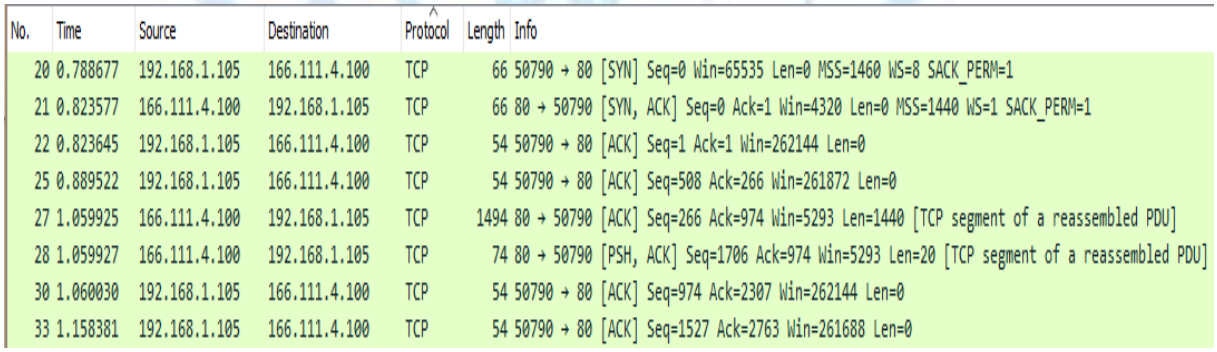


4.2WireShark 所抓取的 TCP 包



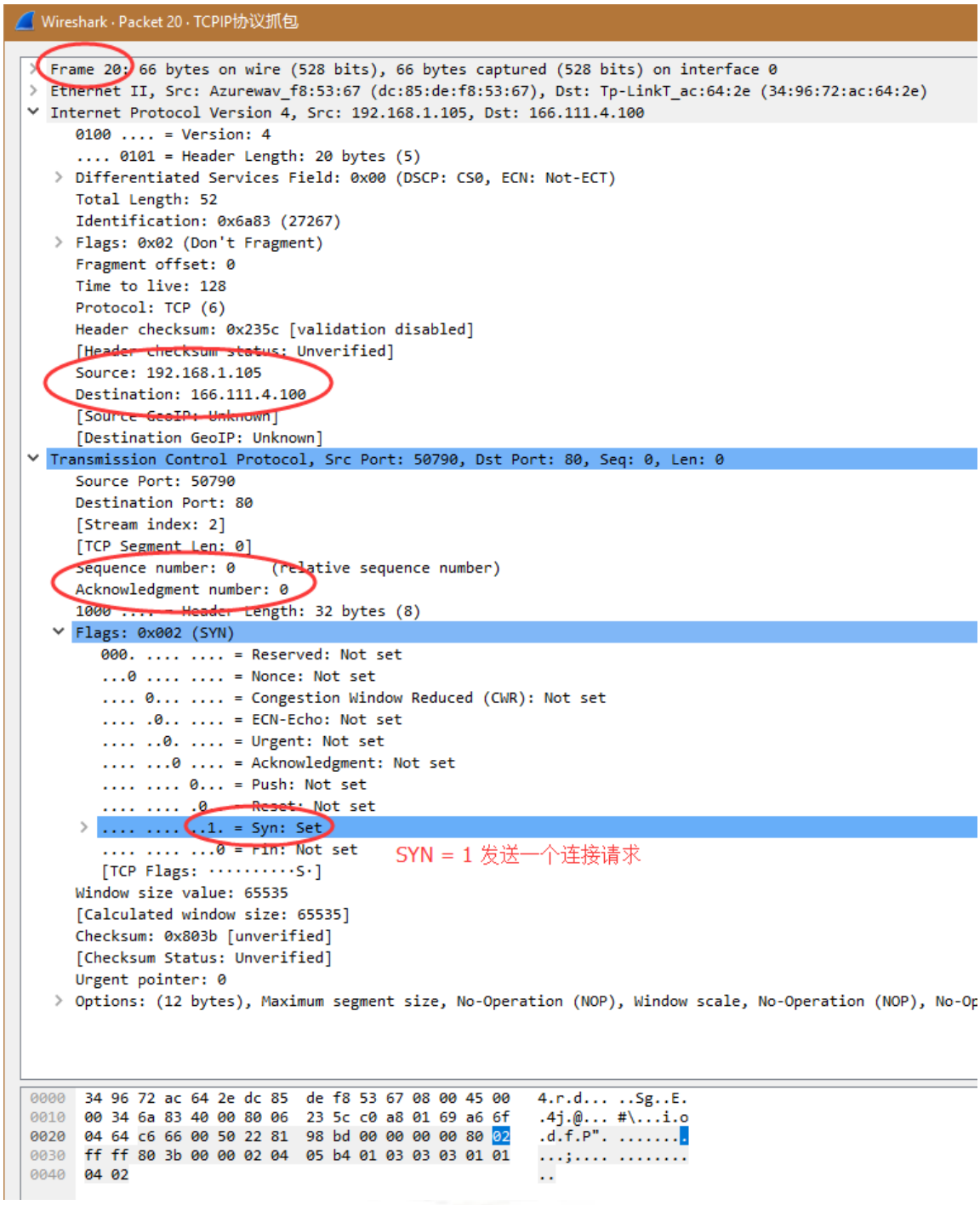


4.2.1 三次握手

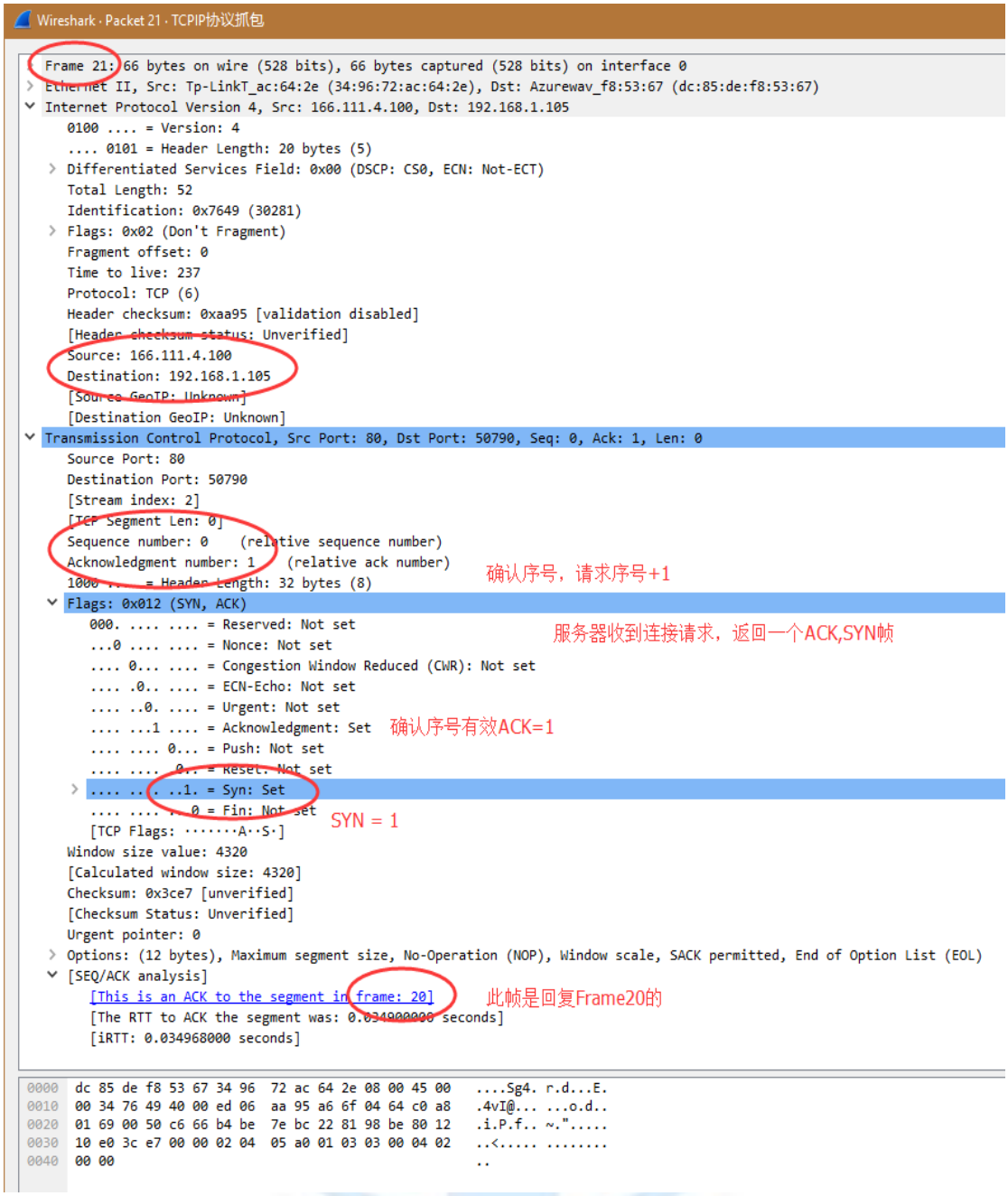


本机 IP 为 192.168.1.105；清华大学官网 IP 为 166.111.4.100。

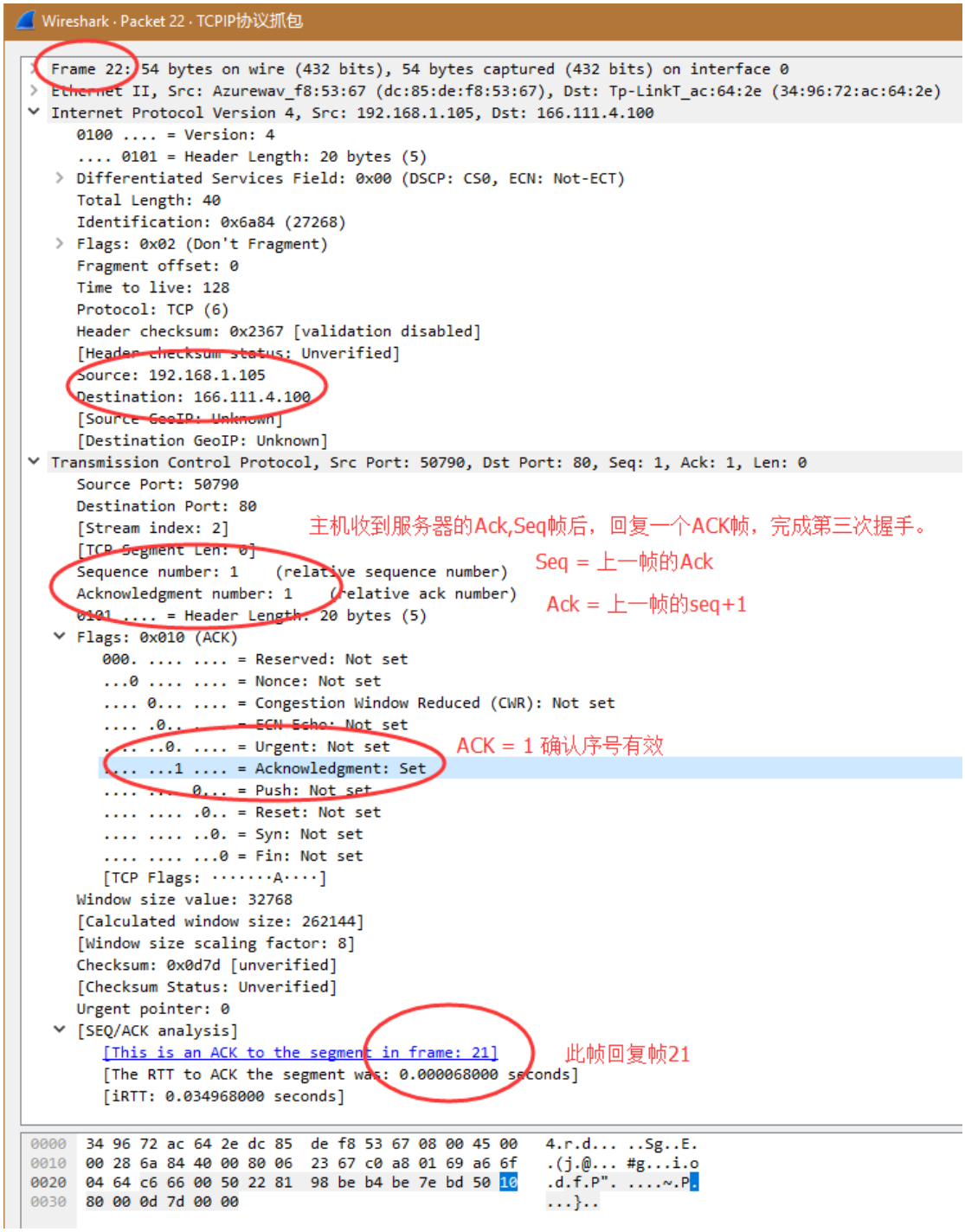
1、主机发起一个 TCP 连接请求（TCP）。



2.、服务器响应连接请求(TCP)。



3、主机返回 ACK 完成 3 次握手成功建立连接（TCP）。



4.2.2 四次挥手

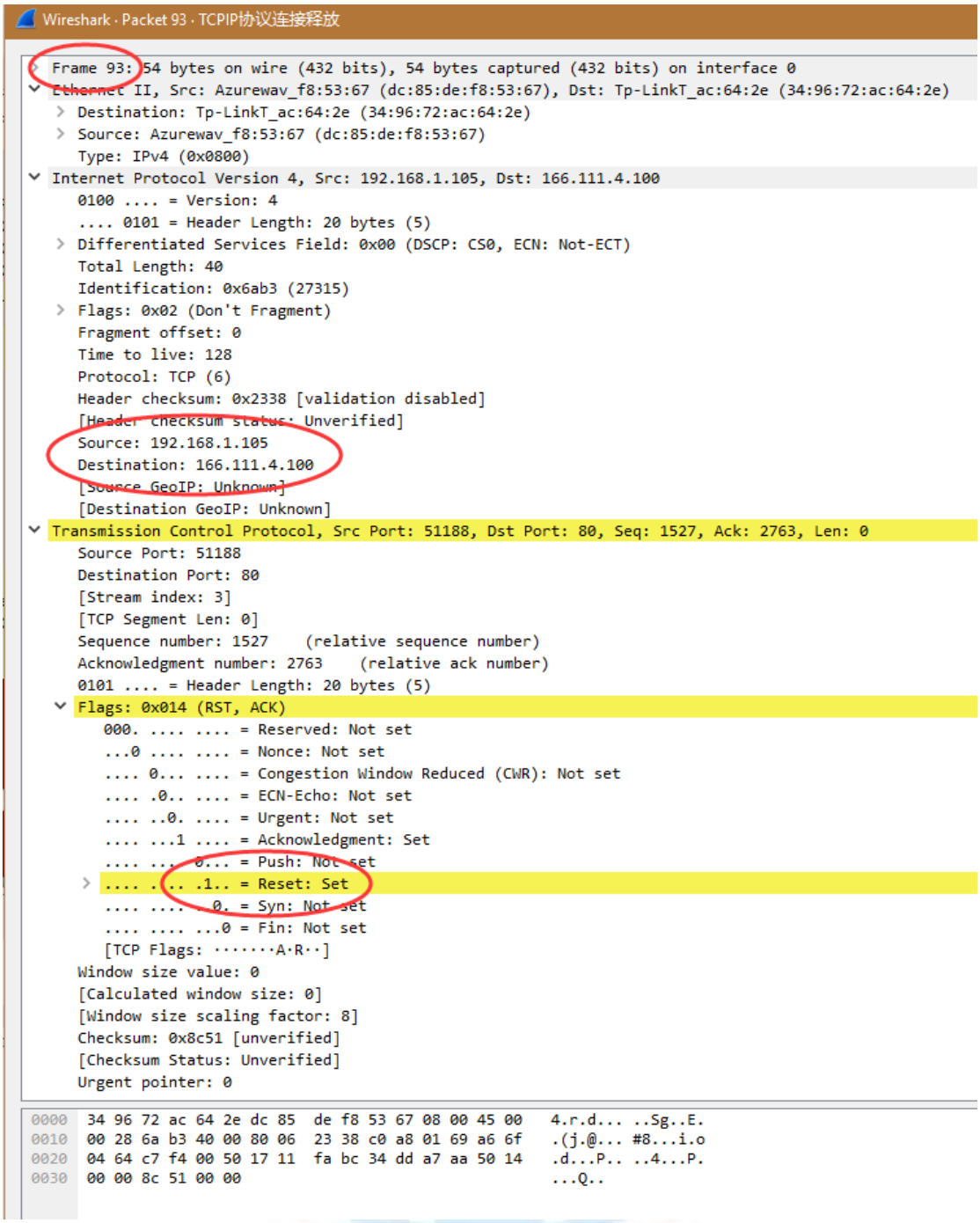
No.	Time	Source	Destination	Protocol	Lengt	Info
74	4.312462	192.168.1.105	166.111.4.100	TCP	54	51188 → 80 [ACK] Seq=1527 Ack=2763 Win=261688 Len=0
93	18.316578	192.168.1.105	166.111.4.100	TCP	54	51188 → 80 [RST, ACK] Seq=1527 Ack=2763 Win=0 Len=0

在 TCP 协议中 RST 表示复位，用来异常的关闭连接，在 TCP 的设计中它是不可或缺的。发送 RST 包关闭连接时，不必等缓冲区的包都发出去，直接就丢弃缓存区的包发送 RST 包。而接收端收到 RST 包后，也不必发送 ACK 包来确认。

Wireshark · Packet 74 · TCPIP协议连接释放

```
> Frame 74: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: Azurewav_f8:53:67 (dc:85:de:f8:53:67), Dst: Tp-LinkT_ac:64:2e (34:96:72:ac:64:2e)
v Internet Protocol Version 4, Src: 192.168.1.105, Dst: 166.111.4.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x6ab2 (27314)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x2339 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.105
    Destination: 166.111.4.100
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
v Transmission Control Protocol, Src Port: 51188, Dst Port: 80, Seq: 1527, Ack: 2763, Len: 0
    Source Port: 51188
    Destination Port: 80
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 1527    (relative sequence number)
    Acknowledgment number: 2763    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  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····]
    Window size value: 32711
    [Calculated window size: 261688]
    [Window size scaling factor: 8]
    Checksum: 0x0c8e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 73]
      [The RTT to ACK the segment was: 0.000083000 seconds]
      [iRTT: 0.039350000 seconds]
```

```
0000  34 96 72 ac 64 2e dc 85  de f8 53 67 08 00 45 00   4.r.d... ..Sg..E.
0010  00 28 6a b2 40 00 80 06  23 39 c0 a8 01 69 a6 6f   .(j.@... #9...i.o
0020  04 64 c7 f4 00 50 17 11  fa bc 34 dd a7 aa 50 10   .d...P.. ..4...P.
0030  7f c7 0c 8e 00 00                                 ......
```



5 参考文献

[1] 《计算机网络》，清华大学出版社，第 5 版，Andrew S.Tanebaum, David J. Wetherall 著，严伟，潘爱民译。

[2] 《计算机网络》,，电子工业出版社，第 7 版，谢希仁著。

[3] 《网络工程技术与实验教程》，清华大学出版社，第 2 版，张新有，袁霞，贾真著。