

# 计算机网络与应用

## 实验一 利用 Wireshark 分析网络协议

学 院： 吴健雄学院

专 业： 人工智能

学 号： 61523233

姓 名： 黄笃修

## 一、实验目的

1. 加强对计算机网络通信协议的理解，用理论知识解决实际问题。
2. 学习 Wireshark 的基本操作，抓取和分析有线局域网的数据包，熟悉一些应用层命令和协议。
3. 通过运用 Wireshark 对网络活动进行分析，观察 TCP 协议报文，分析通信时序，理解 TCP 的工作过程，掌握 TCP 工作原理与实现；学会运用 Wireshark 分析 TCP 连接管理、流量控制和拥塞控制的过程，发现 TCP 的性能问题。

## 二、实验任务

### （一）应用层协议分析

1. 利用 Wireshark 抓包软件，抓取网络数据包；
2. 学习基本操作：捕获过滤器和显示过滤器；
3. 分析 HTTP 和 DNS 协议。

### （二）传输层协议分析

1. TCP 数据流的追踪；
2. TCP 连接的建立；
3. TCP 连接的终止；
4. TCP 连接的重置。

## 三、实验内容（含解析）

（依次描述每个任务的完成过程：包括执行命令语句、过程截图等）

### （一）应用层协议分析

捕获数据包之后使用过滤器留下 http 协议的数据包：

http\_record.pcapng

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(T) 无线(W) 工具(I) 帮助(H)

http

No.	Time	Source	Destination	Protocol	Length	Info
16089	28.466455	10.208.181.238	36.155.107.6	HTTP	674	GET / HTTP/1.1
16093	28.478321	36.155.107.6	10.208.181.238	HTTP	513	HTTP/1.1 301 Moved Permanently (text/html)

## 第一个 HTTP 请求数据包分析：

```
> Frame 16089: 674 bytes on wire (5392 bits), 674 bytes captured (5392 bits) on interface \Device\NPF_{3858E7CE-B57A-471B-B93C-C75CFA40088}, id 0
> Ethernet II, Src: CloudNetwork_2d:0e:99 (44:fa:66:2d:0e:99), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
> Internet Protocol Version 4, Src: 10.208.181.238, Dst: 36.155.107.6
> Transmission Control Protocol, Src Port: 62333, Dst Port: 80, Seq: 1, Ack: 620
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: www.china.com\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\n
  > Cookie: Hm_lvt_cbec92dec763e6774898d6d85460f707=1745913925; Hm_lpvt_cbec92dec763e6774898d6d85460f707=1745913925; HMAccount=483E8516921859E8; area_name=%E6%81%B9%E8%80%B%E7%9-
    \r\n
  [Response in frame: 16093]
  [Full request URI: http://www.china.com/]

```

协议为 HTTP/1.1，源 IP 为 10.208.181.238，目标 IP 为 36.155.107.6，源端口为 62333，目标端口为 80。请求方法是 GET，表示客户端希望从服务器获取指定的资源。User-Agent 标明了客户端的操作系统和浏览器版本，此处为请求来源于一个运行 Windows10/11 的 Chrome 浏览器。Accept 标明了客户端希望接收的内容类型，此处为 HTML、XHTML、图片等格式。Accept-Encoding 表明了客户端支持的压缩格式。Accept-Language 表明了客户端的偏好语言，此处为中文。Upgrade-Insecure-Requests:1 表示浏览器希望通过升级 HTTP 为 HTTPS。

## 第二个 HTTP 响应数据包分析：

```
> Frame 16093: 513 bytes on wire (4104 bits), 513 bytes captured (4104 bits) on interface \Device\NPF_{3858E7CE-B57A-471B-B93C-C75CFA40088}, id 0
> Ethernet II, Src: IETF-VRRP-VRID_01 (00:00:5e:00:01:01), Dst: CloudNetwork_2d:0e:99 (44:fa:66:2d:0e:99)
> Internet Protocol Version 4, Src: 36.155.107.6, Dst: 10.208.181.238
> Transmission Control Protocol, Src Port: 80, Dst Port: 62333, Seq: 1, Ack: 621, Len: 459
v Hypertext Transfer Protocol
  > HTTP/1.1 301 Moved Permanently\r\n
    Server: nginx\r\n
    Date: Tue, 29 Apr 2025 08:05:35 GMT\r\n
    Content-Type: text/html\r\n
  > Content-Length: 162\r\n
    Connection: keep-alive\r\n
    Location: https://www.china.com/\r\n
    x-via: 1.1 PS-WUX-01fPh24:30 (Cdn Cache Server V2.0)\r\n
    x-ws-request-id: 6810884f_PS-WUX-01fPh24_5045-31538\r\n
    \r\n
  [Request in frame: 16089]
  [Time since request: 0.011866000 seconds]
  [Request URI: /]
  [Full request URI: http://www.china.com/]
  File Data: 162 bytes
v Line-based text data: text/html (7 lines)
  <html>\r\n
  <head>\r\n
  <title>301 Moved Permanently</title></head>\r\n
  <body>\r\n
  <center><h1>301 Moved Permanently</h1></center>\r\n
  <hr><center>nginx</center>\r\n
  </body>\r\n
  </html>\r\n

```

协议为 HTTP/1.1，源 IP 为 36.155.107.6，目标 IP 为 10.208.181.238，源端口为 80，目标端口为 62333。响应信息为 HTTP 301 Moved Permanently，即请求

的资源已经永久性地移动到新的位置。状态码：301 表示请求的资源已经被永久移动到新的位置，客户端应根据响应头中的 Location 字段重定向到新的 URL。Date 为服务器发送响应的时间。Content-Type 表示响应内容的格式，此处为 HTML 文本格式。Content-Length 是响应体的长度。Connection: keep-alive 表示服务器希望保持与客户端的连接。Location: https://www.china.com/ 是新的资源位置，客户端应根据该 URL 进行重定向。X-Via: 1. PS-HUX-DCD，表示响应经过了某个代理服务器的处理。Line-based text data 是返回的 html 文件，此处内容就是 301。

(二) 传输层协议分析

使用 tcp.flags.syn == 1 && tcp.flags.ack == 1 过滤表达式过滤出 TCP 数据包后，右击追踪流追踪改会话的所有数据包，观察前三个包的建立连接的握手过程：

tcp.stream eq 11						
No.	Time	Source	Destination	Protocol	Length	Info
4529	20.284926	10.208.181.238	120.241.40.14	TCP	66	64750 → 14563 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
4579	20.324357	120.241.40.14	10.208.181.238	TCP	66	14563 → 64750 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS=1024
4580	20.324396	10.208.181.238	120.241.40.14	TCP	54	64750 → 14563 [ACK] Seq=1 Ack=1 Win=65280 Len=0

数据包 1 为发起连接请求的 TCP SYN 数据包，源 IP 为 10.208.181.238，目标 IP 为 120.241.40.14。数据包 2 为对连接请求的响应，SYN+ACK，表示目标 IP 120.241.40.14 已经收到并同意连接。数据包 3 为连接确认，ACK，表示源 IP 10.208.181.238 确认收到了对方的连接请求。这三个数据包展示了典型的 TCP 三次握手过程：第一次 SYN 请求、第二次 SYN-ACK 响应、第三次 ACK 确认建立连接。

使用 tcp.flags.fin == 1 过滤表达式过滤出 TCP 数据包后，右击追踪流追踪改会话的所有数据包，观察最后四个包的终止连接的挥手过程：

5086	22.924435	10.208.181.238	120.240.153.16	TCP	54	64285 → 14563 [FIN, ACK] Seq=35 Ack=132 Win=255 Len=0
5090	22.964149	120.240.153.16	10.208.181.238	TCP	60	14563 → 64285 [ACK] Seq=132 Ack=35 Win=64 Len=0
5091	22.966131	120.240.153.16	10.208.181.238	TCP	60	14563 → 64285 [FIN, ACK] Seq=132 Ack=36 Win=64 Len=0
5092	22.966146	10.208.181.238	120.240.153.16	TCP	54	64285 → 14563 [ACK] Seq=36 Ack=133 Win=255 Len=0

数据包 1 为源 IP 10.208.181.238 向目标 IP 120.240.153.16 发送了一个 FIN 请求，表示关闭连接。数据包 2 为目标 IP 120.240.153.16 向源 IP 10.208.181.238 发送了 FIN + ACK 响应，确认关闭连接。数据包 3 为目标 IP 120.240.153.16 发送了

一个 ACK 确认包，确认接收了源 IP 10.208.181.238 的关闭请求。数据包 4 为源 IP 10.208.181.238 发送了一个 ACK 确认包，表示已经接收到目标 IP 的关闭响应。这四个数据包展示了 TCP 连接的四次挥手过程：首先发起关闭请求，然后确认对方的关闭请求，并最终完成连接的关闭。

使用 `tcp.flags.reset == 1` 过滤表达式过滤出 TCP 数据包后，右击追踪流追踪改会话的所有数据包，观察连接的重置：

5307	23.794334	10.208.181.238	120.240.153.16	TCP	54 64624 → 14563 [FIN, ACK] Seq=35 Ack=132 Win=255 Len=0
5308	23.795134	120.240.153.16	10.208.181.238	TCP	60 14563 → 64624 [FIN, ACK] Seq=132 Ack=1 Win=64 Len=0
5309	23.795147	10.208.181.238	120.240.153.16	TCP	54 64624 → 14563 [ACK] Seq=36 Ack=133 Win=255 Len=0
5313	23.825828	120.240.153.16	10.208.181.238	TCP	60 14563 → 64624 [RST] Seq=132 Win=0 Len=0
5314	23.825828	120.240.153.16	10.208.181.238	TCP	60 14563 → 64624 [RST] Seq=132 Win=0 Len=0
5315	23.825828	120.240.153.16	10.208.181.238	TCP	60 14563 → 64624 [RST] Seq=133 Win=0 Len=0[Malformed Packet]

最后一个 RST 数据包被标记为“格式错误”，这可能意味着该数据包在传输过程中损坏，或者存在协议错误。这几个数据包显示了连接的异常终止和重置过程，尤其是 RST 标记的数据包，通常用于强制中断连接或在发生错误时关闭连接。

## 四、实验总结

通过这次实验，我学会了抓包软件的使用方法，同时更详细了了解了 HTTP，TCP 协议的内容与过程，通过抓包的方式实际观察到了网络层、传输层协议的运作过程。