第一单元

网络抓包与协议分析

本单元实验要求学生能熟练使用网络抓包软件，捕捉和分析网络数据包，掌握以太网、ARP、IP、ICMP 和 TCP 等重要协议传输单元的结构，深入理解相关网络命令和重要协议算法的工作原理，从而培养学生网络故障检测、网络性能改进和网络安全分析的能力。

实验 1.1 Wireshark 软件使用与 ARP 协议分析

一． 实验目的

学习 Wireshark 的基本操作，抓取和分析有线局域网的数据包；掌握以太网 MAC 帧的基本结构，掌握 ARP 协议的特点及工作过程。

二． 实验内容

使用 Wireshark 抓取局域网的数据包并进行分析：

1. **学习 Wireshark 基本操作：**重点掌握捕获过滤器和显示过滤器。
2. **观察 MAC 地址：**了解 MAC 地址的组成，辨识 MAC 地址类型。
3. **分析以太网帧结构：**观察以太网帧的首部和尾部，了解数据封装成帧的原理。
4. **分析 ARP 协议：**抓取 ARP 请求和应答报文，分析其工作过程。

三． 实验原理

3.1 Wireshark 简介

Wireshark 软件是目前全球使用最广泛的开源网络数据包分析工具（前身为 Ethereal），由 Gerald Combs 编写并于 1988 年获开源许可发布。网络数据包分析是指进入网络通信系统、捕获和解码网络上实时传输数据以及搜集统计信息的过程。通过 Wireshark 对网络数据进行分析，我们能够了解网络是如何运行的、数据包是如何被转发的、应用是如何被访问的；能够分析各层网络协议的性能、掌握通信主体的运行情况，确认带宽分配和时延大小、查看应用的快慢并改进优化，识别网络中存在的攻击或恶意行为、解决网络异常和故障。Wireshark 可以在 Windows、Linux 和 macOS 操作系统中运行，具备友好的图形界面、丰富的统计及图表分析功能。

3.2 以太网 MAC 帧格式

本实验基于使用最广泛的有线局域网（以太网 Ethernet II），以太网的帧结构如表1.1–1所示。其中，MAC 地址（Media Access Control Address，媒体存取控制位址）或

称物理地址（Physical Address），用于在网络中标识网卡。MAC 地址的长度为 48 位 (6 个字节)，通常表示为 12 个 16 进制数，如：00-16-EA-AE-3C-40。其中前 3 个字节的 16 进制数 00-16-EA 代表网络硬件制造商的编号、即组织唯一标志符 (OUI)，它由 IEEE 分配；而后 3 个字节的 16 进制数 AE-3C-40 代表该制造商所生产的某个网络产品 (如网卡) 的系列号。

表 1.1–1 以太网帧格式

前导字符	目的 MAC 地址	源 MAC 地址	类型	IP 数据报	帧校验
8 字节	6 字节	6 字节	2 字节	46-1500 字节	4 字节

3.3 ARP 协议及数据报格式

地址解析协议（Address Resolution Protocol，ARP），主要作用是将 IP 地址解析为 MAC 地址。当某主机或网络设备要发送数据给目标主机时，必须知道对方的网络层地址（即 IP 地址），而且在数据链路层封装成帧时，还必须有目标主机（或下一跳路由器）的 MAC 地址。本实验重点观察最简单的情形：同一个网段内，主机 A 要向主机 B 发送信息时，ARP 解析的过程（主机 A 和 B 不在同一网段的情况请参阅课本相关内容）。具体如下：

1. 主机 A 首先查看自己的 ARP 表。如果找到了主机 B 的 MAC 地址，则利用这个地址对 IP 数据报进行帧封装，并将数据报发送给主机 B。
2. 如果主机 A 在 ARP 表中找不到主机 B 的 MAC 地址，则以广播方式发送一个 ARP 请求报文。ARP 请求报文中的发送端 IP 地址和发送端 MAC 地址为主机 A 的 IP 地址和 MAC 地址，目标 IP 地址和目标 MAC 地址为主机 B 的 IP 地址和全 0 的 MAC 地址。由于 ARP 请求报文以广播方式发送，该网段上的所有主机都可以接收到该请求，但只有被请求的主机 B 会对该请求进行处理。
3. 主机 B 比较自己的 IP 地址和 ARP 请求报文中的目标 IP 地址，当两者相同时进行如下处理：将 ARP 请求报文中的发送端（即主机 A）的 IP 地址和 MAC 地址存入自己的 ARP 表中。然后以单播方式发送 ARP 响应报文给主机 A，其中包含了自己的 MAC 地址。

4. 主机 A 收到 ARP 响应报文后，将主机 B 的 MAC 地址加入到自己的 ARP 表中以用于后续报文的转发，同时将 IP 数据报进行封装后发送出去。

ARP 报文结构如图1.1–1所示，ARP 报文总长度为 28 字节，MAC 地址长度为 6 字节，IP 地址长度为 4 字节。每个字段的含义如下：

- **硬件类型：**指明了发送方想知道的硬件接口类型，以太网的值为 1。
- **协议类型：**表示要映射的协议地址类型。IP 地址的类型值为 0x0800。
- **硬件地址长度和协议地址长度：**分别指出硬件地址和协议地址的长度，以字节为单位。在以太网中，它们的值分别为 6 和 4。
- **操作码（op）：**用来表示这个报文的类型，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP 响应为 4。

图 1.1–1 ARP 报文结构示意图

3.4 实验方法及手段

使用 Wireshark 软件在有线局域网中捕捉相关网络操作的数据包，运用观察对比、计算验证、分析统计等方法，掌握以太网 MAC 帧和 IP 数据报的结构以及 ARP 协议的工作原理。

四． 实验条件

- PC 机一台，连入局域网；

- Wireshark 软件，建议 3.0 以上版本。

五． 实验步骤

5.1 WireShark 基本使用

1. 通过 Wireshark 官网下载最新版软件，按默认选项安装。
2. 运行 Wireshark 软件，程序界面会显示当前的网络接口列表，双击要观察的网络接口，开始捕捉数据包，Wireshark 软件选择网络接口的界面如图1.1–2所示。

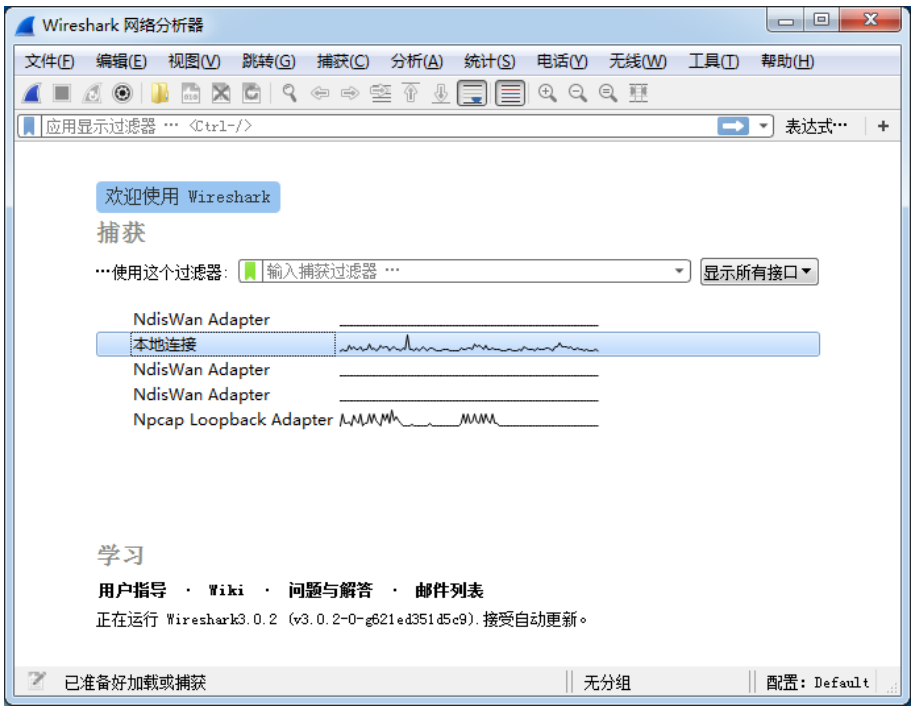

图 1.1–2 Wireshark 软件启动界面

3. 点击工具栏上的红色方块按钮停止捕捉。
4. 菜单、工具栏、状态栏和主窗口如图1.1–3所示，可以根据需要通过菜单“视图”以及“编辑/首选项/外观”的相关选项对基本设置进行更改。例如图1.1–4中的语言、字体缩放、颜色、布局等项目。
5. 使用“显示过滤器”可以方便地从捕获的数据包中筛选出要观察的数据包。显示过滤器支持若干的过滤选项：源 MAC、目的 MAC、源 IP、目的 IP、TCP/UDP 传输协议、应用层协议（HTTP, DHCP）、源端口 Port、目的端口 Port 等。在显示

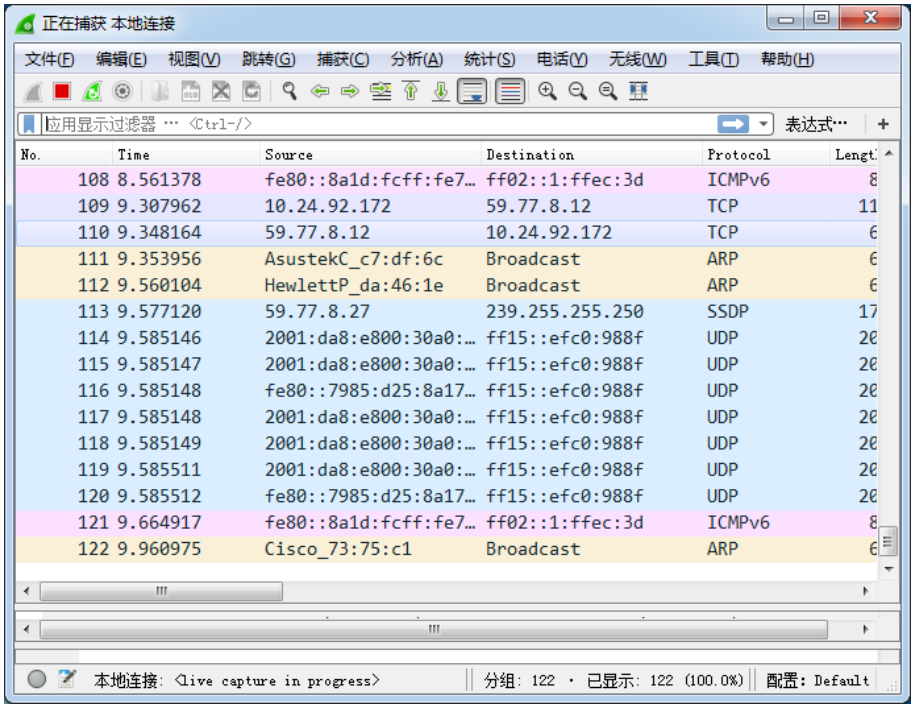

图 1.1–3 Wireshark 主窗口界面

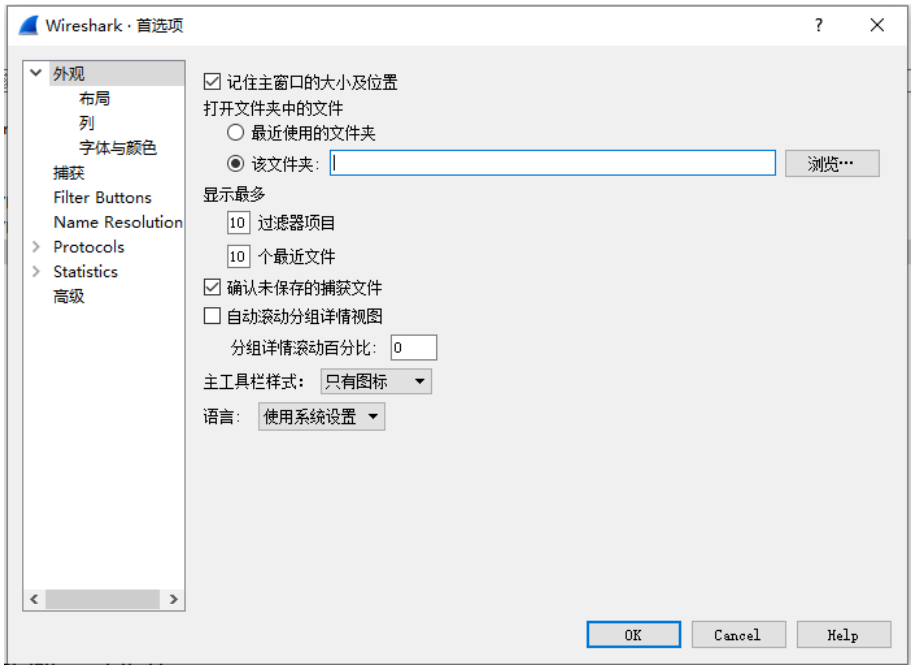

图 1.1–4 Wireshark 的设置界面

过滤器栏中输入过滤表达式（更详细的显示过滤语法可以查看 WireShark 的官方文档[1]），例如下面的命令：

- `arp` //显示 arp 协议报文，例如图1.1--5
- `ip.src == a.b.c.d && icmp` //显示源地址为 a.b.c.d 的 icmp 报文

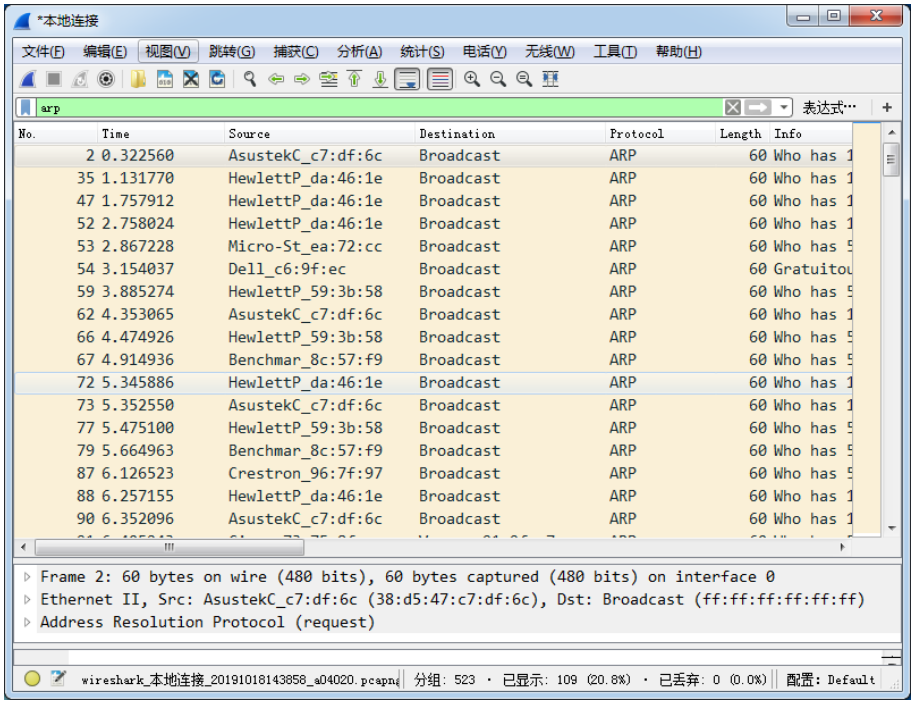

图 1.1–5 显示过滤规则的示例

6. 通过主菜单“文件”/“导出特定分组”（如图1.1–6），可以保存捕获的网络数据（也可以先选中某些包，只保存部分数据）。
7. 如果只想捕捉特定的数据包，可以使用菜单“捕获”/“捕获过滤器”选定想要的类型（如图1.1–7）。例如，选择“IPv4 only”，Wireshark 只抓取 ipv4 类型的数据包。Wireshark 过滤器官方文档提供了更加全面详细的语法和常用示例[2]。
8. Wireshark 还提供了丰富的统计功能供用户选用，如图1.1–8。更多文档可以查询 Wireshark 使用帮助[3]。

[1] WireShark 显示过滤器语法
[2] WireShark 常用过滤器语法
[3] WireShark 学习手册

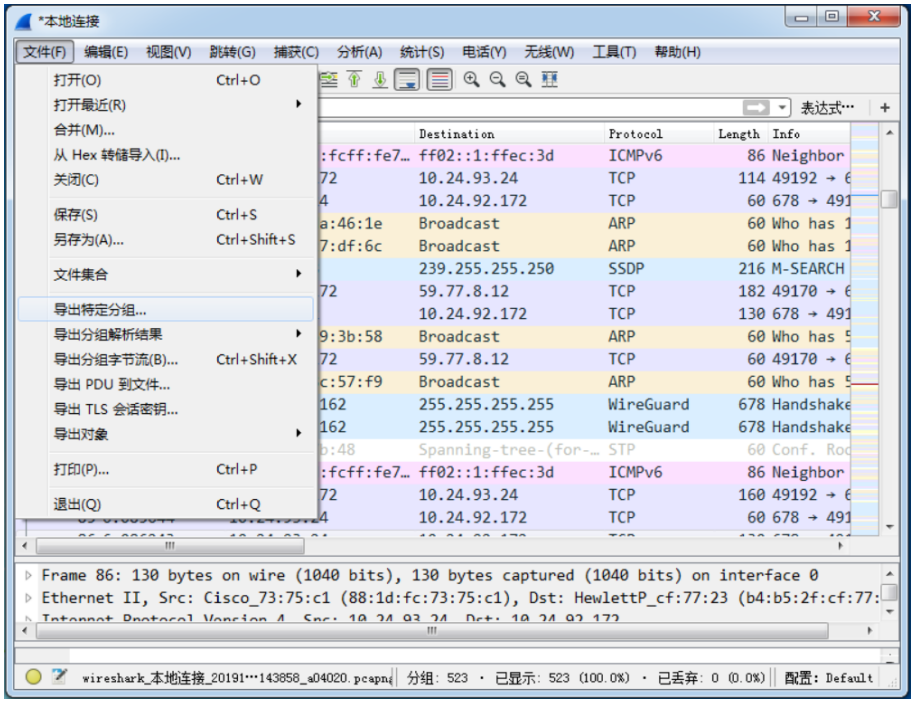

图 1.1–6 操作主菜单保存数据文件

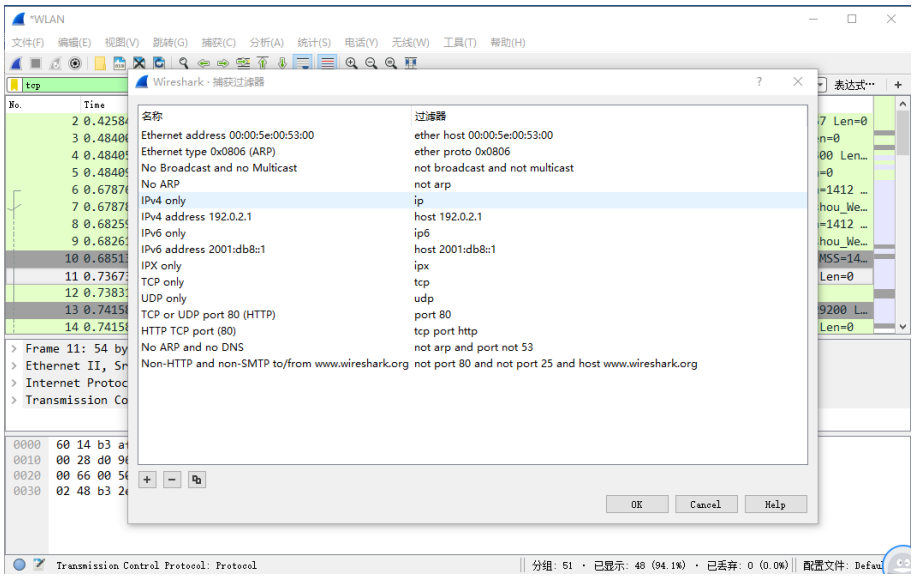

图 1.1–7 选中特定的捕获类型

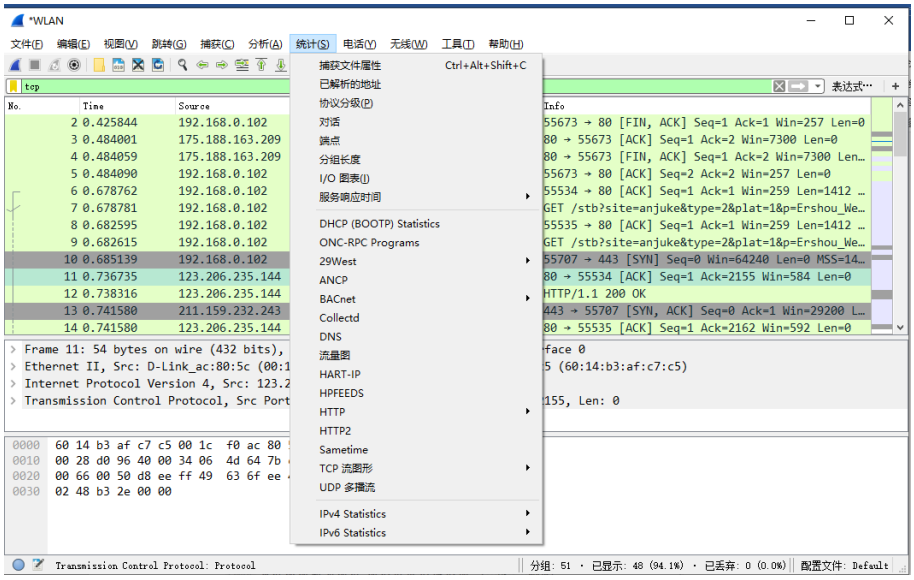

图 1.1–8 统计功能

5.2 观察 MAC 地址

启动 Wireshark 捕捉数据包，在命令行窗口分别 ping 网关和 ping 同网段的一台主机，分析本机发出的数据包。重点观察以太网帧的 Destination 和 Source 的 MAC 地址，辨识 MAC 地址类型，解读 OUI 信息、I/G 和 G/L 位。

5.3 分析以太网的帧结构

选择其中一个数据包，点击 Ethernet II 展开（图1.1–9），查看 MAC 帧的各个字段。

5.4 ARP 协议分析

1. 使用 `arp -d` 命令（其语法见图1.1–10），清空本机的 ARP 缓存，开启 Wireshark，ping 本机的同网段地址，在显示过滤器条框中输入“`arp`”，观察捕获的 ARP 报文的各个字段，分析请求/响应的过程。
2. 使用 `arp -d` 命令，清空本机的 ARP 缓存。开启 Wireshark，ping 与本机网段不同的 IP 地址或域名，观察捕获的 ARP 报文的各个字段，分析请求/响应的过程。

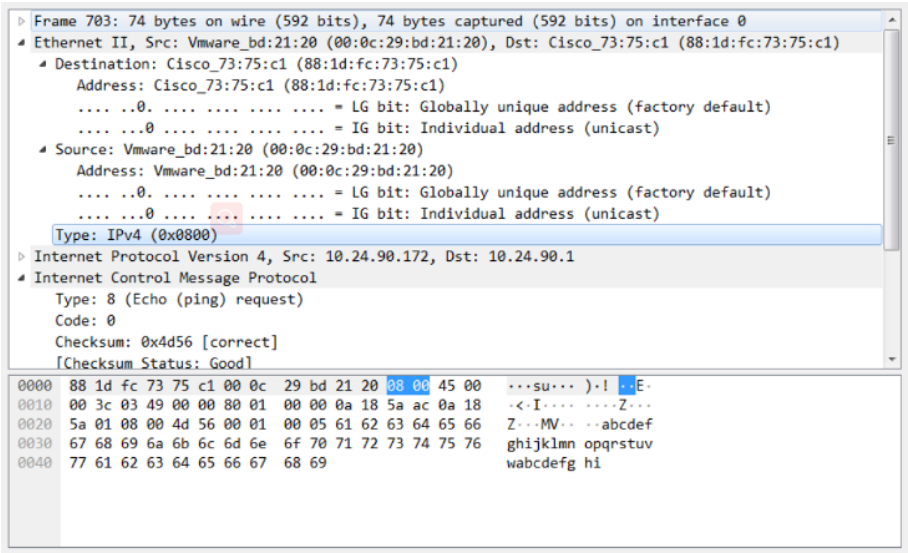

图 1.1–9 以太网帧结构展开界面

```
arp [-a [InetAddr] [-N IfaceAddr]] [-g [InetAddr] [-N IfaceAddr]]
    [-d InetAddr [IfaceAddr]] [-s InetAddr EtherAddr [IfaceAddr]]

  -a 显示所有接口/特定接口的当前 ARP 缓存表
  -g 同 -a
  -d 删除所有/指定的 IP 地址项
  -s 在 ARP 缓存中添加对应 InetAddr 地址的 EtherAddr 地址静态项
```

图 1.1–10 arp 命令语法及参数

六． 思考题

1. 使用了显示过滤器后，Wireshark 的抓包工作量会减少吗？
2. MAC 帧的长度和 IP 数据报的长度有怎样的关系？请用你的数据记录进行验证。
3. 假设本机 IP 地址是 192.168.0.38，在本机上运行 Wireshark 捕获报文，使用“`ip.addr == 192.168.0.38`”作为过滤条件，能否过滤出本机发出/收到的 ARP 报文？为什么？
4. ping 同一局域网内的主机和局域网外的主机，都会产生 ARP 报文么？所产生的 ARP 报文有何不同，为什么？

5. ARP 请求数据包是支撑 TCP/IP 协议正常运作的广播包。如果滥发或错发 ARP 广播包会产生那些不良影响？如何发现和应对？
6. 什么是免费 ARP（Gratuitous ARP）？它的作用是什么？请使用 Wireshark 进行捕捉和分析。

七． 考核方法

考核分为现场考核和实验报告两个部分，具体如下。报告内容应包含以下内容，相关的分析解释都需要截图证明，并与提交的 Wireshark 抓包数据文件相吻合。

1.（20 分）现场考核：Wireshark 的基本使用。
2.（30 分）以太网帧格式分析：MAC 地址类型、头部信息、长度及封装。
3.（30 分）结合捕捉的网络数据，分析 ARP 数据包，描述 ARP 协议工作过程。
4.（10 分）回答任意两道思考题。
5.（10 分）记录自己在本次实验中所遇到的问题，以及心得感悟。如果遇到异常情况，或者无法完成任务时，也请分析错误产生的原因。

实验 1.2 IP 与 ICMP 分析

一． 实验目的

IP 和 ICMP 协议是 TCP/IP 协议簇中的网络层协议，在网络寻址定位、数据分组转发和路由选择等任务中发挥了重要作用。本实验要求熟练使用 Wireshark 软件，观察 IP 数据报的基本结构，分析数据报的分片；掌握基于 ICMP 协议的 ping 和 traceroute 命令及其工作原理。

二． 实验内容

启动 Wireshark，捕捉网络命令执行过程中本机接受和发送的数据报。

1. **执行 ping 命令，观察 IP 数据报和 ICMP 询问报文的结构：**通过 Wireshark 监视器观察捕获流量中的 ICMP 询问报文和 IP 数据报的结构。注意比较 ICMP 请求帧与回应帧，及其 IP 头部数据字段的异同。
2. **改变 ping 命令的参数，观察 IP 数据报分片：**更改 ping 命令参数 MTU，使其发出长报文以触发 IP 数据报分片，再观察 IP 数据报的结构变化。
3. **执行 Traceroute 命令，观察 ICMP 差错报文的结构，并分析其工作原理：**使用 Linux 操作系统提供的 traceroute 命令（或者 Windows 系统提供的 tracert 命令），捕获和分析该命令所产生的 IP 数据报，特别注意相关的 ICMP 差错报文。结合捕获的具体数据，画出命令执行过程中数据交互的示意图，掌握 traceroute 的工作原理。

三． 实验原理、方法和手段

3.1 IP 协议及数据报格式

网际互连协议（Internet Protocol，IP），是 TCP/IP 体系中的网络层协议，可实现大规模的异构网络互联互通，为主机提供无连接的、尽力而为的数据包传输服务。在网际协议第 4 版（IPv4）中，IP 数据报是一个可变长分组，包括首部和数据两部分 (如图1.2–1)。首部由 20 ~ 60 字节组成，包含与路由选择和传输有关的重要信息，其各字段意义如下：

1. **版本（4 位）：**该字段定义 IP 协议版本，所有字段都要按照此版本的协议来解释。

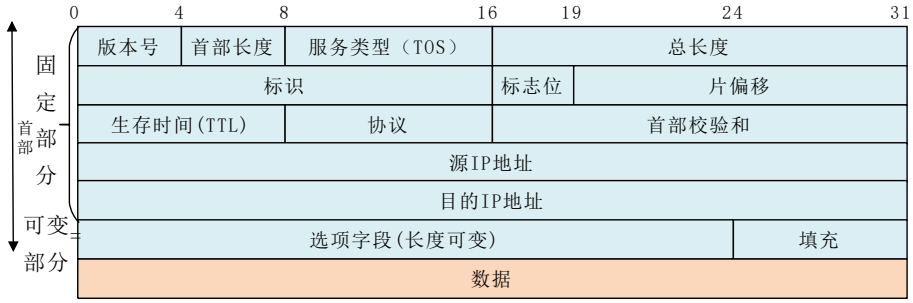

图 1.2–1 IP 数据报结构示意图

2. **首部长度（4 位）：**该字段定义数据报协议头长度，表示协议首部具有 32 位字长的数量，最小值为 5，最大值为 15。
3. **服务（8 位）：**该字段定义上层协议对处理当前数据报所期望的服务质量，并对数据报按照重要性级别进行分配。前 3 位为优先位，后面 4 位为服务类型，最后 1 位没有定义。这 8 位可用于分配优先级、延迟、吞吐量以及可靠性。
4. **总长度（16 位）：**该字段定义整个 IP 数据报的字节长度，包括协议首部和数据，其最大值为 65535 字节。
5. **标识（16 位）：**该字段包含一个整数，用于标识当前数据报。当数据报分片时，标识字段的值被复制到所有的分片中。
6. **标记（3 位）：**该字段由 3 位字段构成，其中最低位（MF）控制分片：若存在下一个分片则值为 1；否则置 0 代表该分片是最后一个。中间位（DF）指出数据报是否可进行分片，若置 1 则不允许该数据报进行分片。第三位即最高位保留不使用，值为 0。
7. **分片偏移（13 位）：**该字段指出数据分片在源数据报中的相对位置，以 8 字节为长度单位。
8. **生存时间（8 位）：**该字段是计数器，转发该数据报的路由器依次减 1 直至减少为 0。
9. **协议（8 位）：**该字段指出在 IP 层处理后，由哪种上层协议接收该数据报。
10. **头部校验和（16 位）：**该字段帮助确保 IP 协议头的正确性。计算过程是先将校验和字段置为 0，然后将整个头部每 16 位划分为一部分，并将各部分相加，其计算结果取反码，填入校验和字段中。

11. **源地址（32 位）：**源主机的 IP 地址。
12. **目的地址（32 位）：**目标主机的 IP 地址。

一个 IP 包从源主机传输到目标主机可能需要经过多个传输媒介不同的网络。每种网络对数据帧都设置了一个最大传输单元 (MTU) 的限制（例如以太网的 MTU 是 1500 字节）。因此，当路由器在转发 IP 包时，如果数据包的大小超过了出口链路网络的 MTU 时，需将对该 IP 数据报进行分片，才能在目标链路上顺利传输。每个 IP 分片将独立传输，直到所有分片都到达目的地后，目标主机才会把他们重组成一个完整的 IP 数据报。在 IP 数据报的分片与重组过程中，以下三个首部字段发挥了重要作用：

1. 标记的后两位：最低位记为 MF（More Fragment），MF = 1 代表还有后续分片，MF = 0 表示此为原始数据报的最后分片。次低位 DF（Don't Fragment），用来控制数据报是否允许分片。DF = 1 表示该数据报不允许分片；DF = 0 允许分片。
2. 标识符：用于目的主机将 IP 数据报的各个分片重装成原来的数据报。
3. 片偏移：以 8 字节为单位，目的主机在重装 IP 数据报时需要根据该字段提供偏移量进行排序。这是因为数据分片的独立传输使各分片的到达顺序难以确定。

3.2 ICMP 协议及报文格式

因特网控制报文协议（Internet Control Message Protocol，ICMP），用于 IP 主机、路由器之间传递控制消息。控制消息是指网络是否连通、主机是否可达、路由是否可用等网络本身的控制管理消息，对网络正常运行起着重要的作用。

ICMP 报文的类型可以分为 ICMP 差错报文和 ICMP 询问报文两种（其结构如图1.2–2）。ICMP 差错报告报文主要有终点不可达、源站抑制、超时、参数问题和路由重定向 5 种。ICMP 询问报文有回送请求和应答、时间戳请求和应答、地址掩码请求和应答以及路由器询问和通告 4 种。其常见的类型与代码如表1.2–1所示。

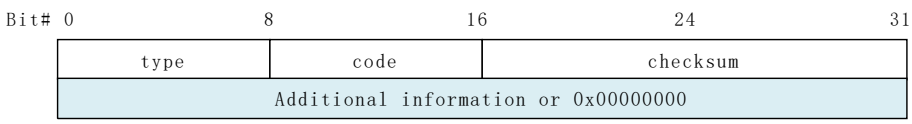

图 1.2–2 ICMP 报文结构示意图

本实验涉及以下两个常用网络命令，都属于 ICMP 协议的典型应用。

表 1.2–1 ICMP 各类型报文的格式

类型 (TYPE)	代码 (CODE)	描述 (Description)	查询类 (Query)	差错类 (Error)
0	0	Echo Reply——回显应答（Ping 应答）	✓	
3	1	Host Unreachable——主机不可达		✓
3	3	Port Unreachable——端口不可达		✓
3	4	Fragmentation needed but no frag. bit set ——需要进行分片但设置不分片比特		✓
8	0	Echo request——回显请求（Ping 请求）	✓	
11	0	TTL equals 0 during transit ——传输期间生存时间为 0		✓

1. ping 命令，是测试网络最有效的工具之一。它是由主机或路由器执行 ping 命令向一个特定的目的主机发送一份 ICMP 回显请求（Echo request）报文，并等待其返回 ICMP 回显应答（Echo Reply）。ping 命令可以检测网络的连通性，简单估测数据报的往返时间（Round Trip Time），确定是否有数据包丢失或损坏，从而帮助分析网络故障。ping 命令格式和常用参数罗列如下：

```
ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos]
     [-r count] [-s count] [-j computer-list] | [-k computer-list]
     [-w timeout] destination-list

    -a 将地址解析为计算机名。
    -n count 发送 count 指定的 ECHO 数据包数。默认值为 4。
    -l length 发送包含由 length 指定的数据量的 ECHO 数据包。
       默认为 32 字节；最大值是 65,527。
    -f 在数据包中发送“不要分片”标志，避免数据包被路由上的网关分片。
    -i ttl 将“生存时间”字段设置为 ttl 指定的值。
```

2. traceroute/tracert 命令，利用 TTL 字段和 ICMP 差错类型报文，查找 IP 数据报的路由转发路径（含路由器信息）。源主机执行该命令向目的主机发送生存时间（TTL）不同的 ICMP 回送请求报文，直至收到目的主机应答，并通过分析应答

报文获得转发路径和时延信息。

首先源主机发起一个 TTL=1 的 ICMP 报文。第一个路由器收到该报文后，TTL 减 1 变为 0 并丢弃此报文，返回一个 [ICMP time exceeded] 的消息。源主机通过这个消息获知 IP 数据报转发路径上的第一个路由器信息。然后，依次增加发送 ICMP 报文的 TTL 值，可以获取路径上的后续路由器的信息。当到达目的地时，目标主机返回一个 [ICMP port unreachable] 的消息，使发起者确认 IP 数据报已经正常到达。至此，traceroute 命令发起者已经获得了通向目标主机路径上的所有路由信息。tracert 命令（Linux）格式和常用参数罗列如下：

```
tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] [-R]
        [-S srcaddr] [-4] [-6] target_name

    -d 将地址解析成主机名
    -h maximum_hops 搜索目标的最大跃点数
    -j host-list 与主机列表一起的松散源路由，用于 IPv4
    -w timeout 等待每个回复的超时时间（单位：毫秒）
    -R 跟踪往返行程路径，用于 IPv6
    -S srcaddr 要使用的源地址，用于 IPv6
    -4 强制使用 IPv4
    -6 强制使用 IPv6
    target_name 指定目标，可以是 IP 地址或主机名
```

3.3 实验方法和手段

1. 使用 Wireshark 软件，捕获本机在 ping 和 traceroute 网络命令执行过程中接收和发出的全部数据流量。
2. 合理设置过滤条件，观察 IP 数据报和 ICMP 报文，着重分析报文首部和内容变化，从而掌握协议的工作原理。
3. 调整 ping 命令的参数，观察并分析 IP 数据报分片情况。
4. 结合所捕获的数据报，画出 traceroute 命令过程中数据交互示意图。

四． 实验条件

装有 Wireshark 软件的 PC 机一台，处于局域网环境。

参考资料：

- J.F Kurose and K.W. Ross, Wireshark Lab: ICMP v8.0
- Wireshark 官方过滤器语法指导书
- IP 协议的 RFC

五． 实验步骤

5.1 ping 命令

本机（示例 IP 为 192.168.1.251）启动 Wireshark 软件，选择要监听的网络接口（如 eth0、wlan0）；然后在终端发起网络命令：ping IP 地址/域名。

1. 在 Wireshark 监视器中设置过滤条件。例如图1.2–3设置过滤条件为 icmp，则显示出所捕获的 ICMP 数据包。

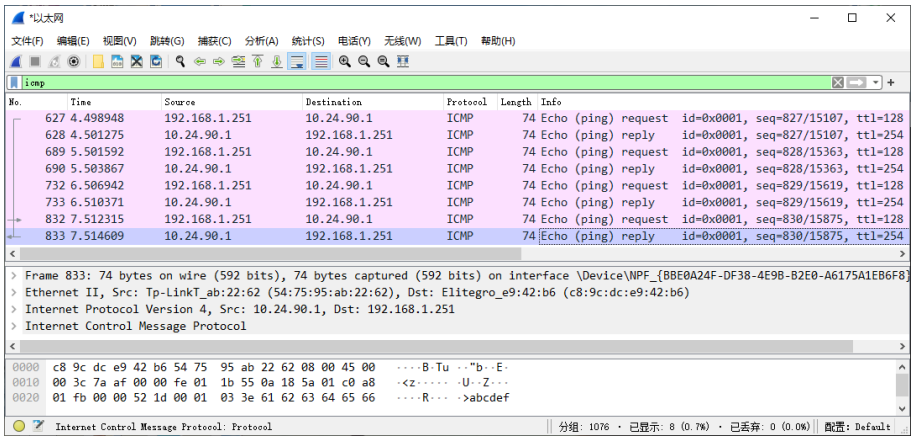

图 1.2–3 Wireshark 监视器界面

2. 点击 Internet Protocol Version 4 展开（如图1.2–4），查看 IP 数据报，特别观察 IP 数据报的首部字段及其内容。
3. 点击 Internet Control Message Protocol 展开（如图1.2–5），查看 ICMP 报文，并解释回显（Echo Request 和 Echo Reply）报文的首部字段。

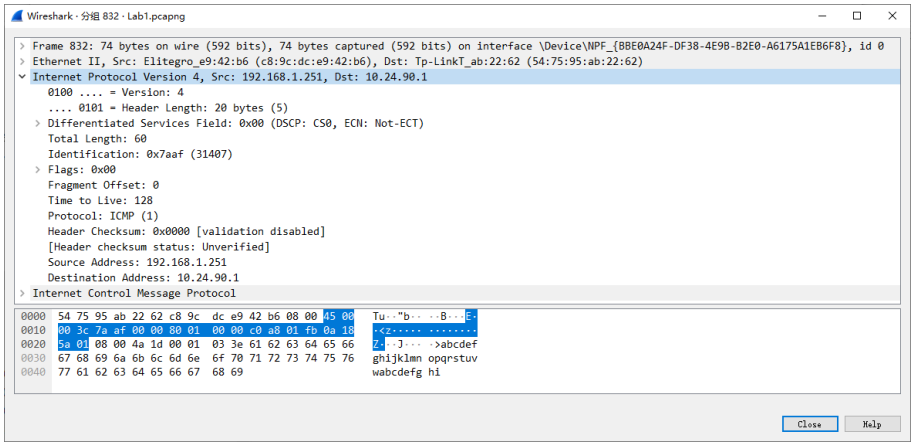

图 1.2–4 查看 IP 数据报

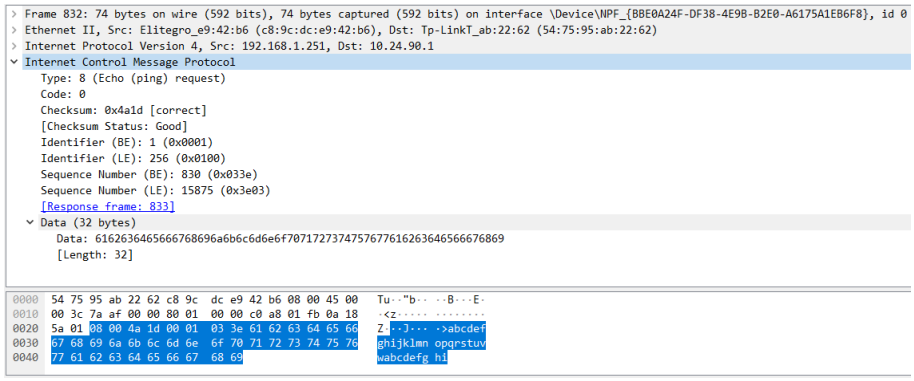

图 1.2–5 Echo request 示例

4. 清空 Wireshark 监控器，重新发起网络命令（如图1.2–6）：ping IP 地址/域名 -l #length，并解释对比前后两次执行 ping 命令的结果。其中，-l #length 确定 echo 数据报的长度为 #length，其默认值为 32 字节，且小于 65,527 字节。
5. 可以多次改变 #length 的大小（例如 1000 字节、2000 字节和 4000 字节），观察 IP 数据报何时会分片？请解释 IP 数据报分片的原因和具体情况。提示：请先确认该网络的 MTU，可在 Wireshark 记录中查找“IPv4 fragments”项目。

5.2 traceroute 命令

本机（示例 IP 为 192.168.1.251）启动 Wireshark 软件，选择要监听的网络接口（如 eth0、wlan0）；然后在终端发起网络命令：traceroute IP 地址/域名。

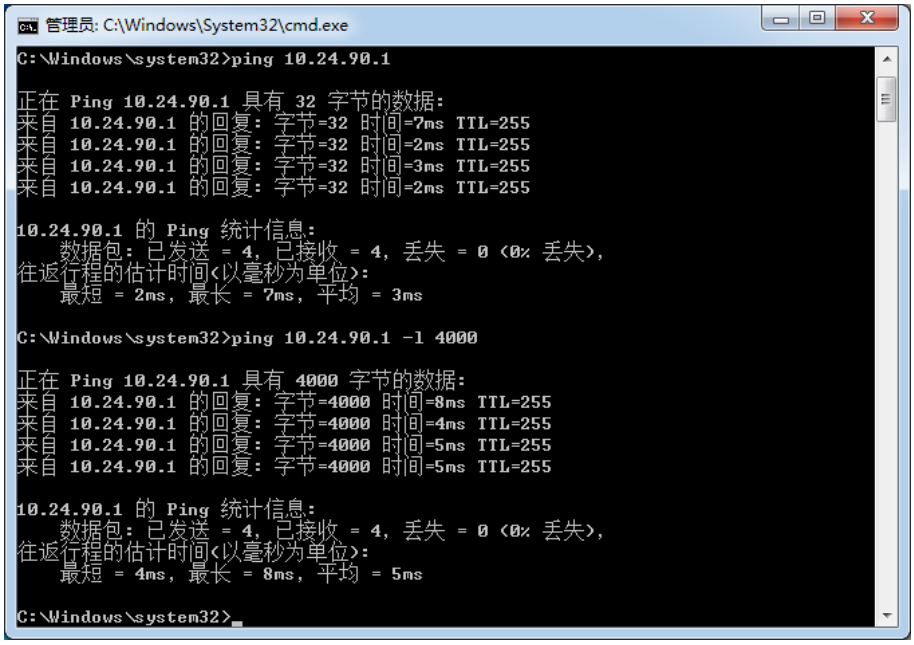

图 1.2–6 ping 命令执行示例

1. 启动 Wireshark 软件，选择要监听的网络接口，设置过滤条件 icmp（如图1.2–7）。

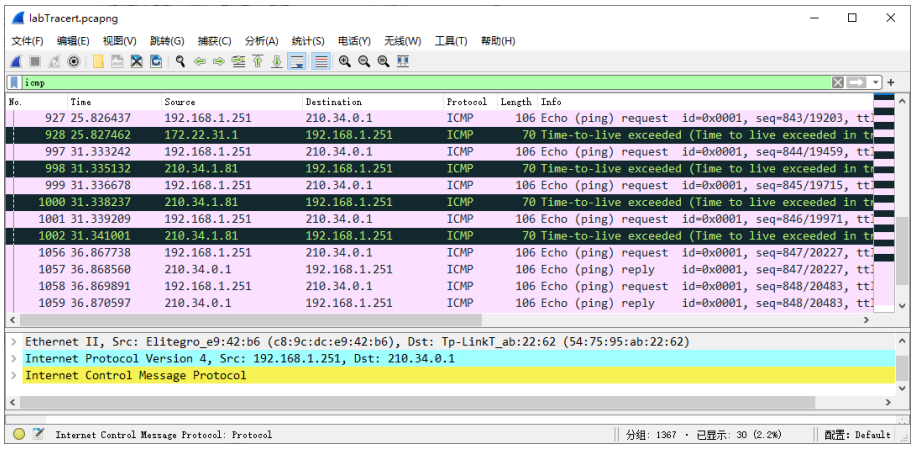

图 1.2–7 在 Wireshark 中设置过滤条件

2. 在终端中使用 traceroute 命令，目的主机是外网的一台设备（如图1.2–8，示例 IP 为 210.34.0.1）。
3. 点击 Internet Control Message Protocol 展开，查看 ICMP 差错报文，观察并解释 ICMP 报文结构和字段内容。
4. 结合 ICMP 报文记录画出数据交互示意图，并描述 tracert 工作原理。

图 1.2–8 在终端中执行 traceroute 命令示例

六． 思考题

1. 在有线局域网中，PC1 的 IP 地址为 192.168.1.5/24，默认路由器的 IP 地址为 192.168.1.1；PC2 的 IP 地址为 192.168.2.6/24，默认路由器的 IP 地址为 192.168.2.1。在 PC1 向 PC2 发送数据的传输过程中，以太网数据帧的首部字段和 IP 数据报的首部字段是怎样变化的？如果有条件，请搭建实验平台进行实验，并使用 Wireshark 软件验证你的答案。
2. 拒绝服务（Denial of Service，DoS）攻击，DoS 通过消耗目标主机设备的某种资源，导致其网络服务不能被正常用户使用。
   (a) IP 数据报分片机制可能被攻击者利用来构建拒绝服务攻击。试设计一种利用 IP 数据报分片机制发动 DoS 攻击的方法，并提出防御的思路。
   (b) 请思考利用 ICMP 报文构建 DoS 攻击的可能性以及防御方法。
3. 在实际操作中，Traceroute 命令返回的某些条目以“*”号表示。请思考有哪些原因可能导致这样的情况。
4. 发送方要怎样决定 IP 数据报分组大小，才能避免因为不同网络 MTU 不一致而引起分片呢？
5. 从 PC1 给 PC2（其地址为 `#IP`）发送三个 ping 命令，请比较命令的结果，并用 Wireshark 软件进行观察分析。
   (a) `ping #IP -l 1472 -f -n 1`
   (b) `ping #IP -l 1473 -f -n 1`
   (c) `ping #IP -l 1473 -n 1`

七． 考核方法

本次实验需提交一份实验报告和一组 Wireshark 数据存储文件。报告内容应当包括以下三个部分，其中的分析解释都有对应的截图，且与数据存储文件记录相符。

1.（30 分）实施 ping 命令，记录引发的 IP 数据报和 ICMP 报文，保存为 pcapng 文件；解释任一个 IP 数据报的首部，并对比 ICMP Echo 请求帧和回应帧；改变 ping 的长度参数，解释 IP 数据报分片情况。
2.（40 分）实施 tracert 命令，记录引发的 ICMP 报文，保存为 pcapng 文件；解释任一个 ICMP 差错报文的结构；描述 tracert 工作原理，结合 ICMP 报文记录画出数据交互示意图。
3.（20 分）回答任意两道思考题。
4.（10 分）记录自己在本次实验中所遇到的问题，以及心得感悟。如果遇到异常情况，或者无法完成任务时，也请分析错误产生的原因。

实验 1.3 TCP 与拥塞控制

一． 实验目的

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。本实验通过运用 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． 实验内容

启动 Wireshark，捕捉网络活动中的 TCP 报文并按要求分析。

1. 连接管理：观察正常 TCP 连接中的三次握手与四次挥手报文，绘制出时序图，并标出双方 TCP 状态变化。
2. 异常情况分析：观察分析 TCP 连接建立过程的异常 (例如，尝试连接未存活的主机或未监听端口，客户端发送了第一个 SYN 连接请求而服务端无响应)；观察 SYN 洪泛影响；观察分析 TCP 通信过程中的各类异常报文 (例如数据超时、乱序)，了解其触发机制与含义。
3. 流量控制 (进阶)：运行一组 TCP 连接客户端/服务器程序，制造收发不平衡场景，观察收发报文中通告窗口的变化，分析与窗口机制相关的类型报文，了解滑动窗口工作原理。
4. 拥塞控制 (进阶)：改变带宽、时延、丢包率等网络参数，观察大文件传输过程，分析识别 TCP 的慢启动、拥塞避免、快速恢复等拥塞控制阶段；在构建的网络试验环境下运行 iperf3 进行网络性能测试，比较不同拥塞控制策略的性能表现。

三． 实验原理、方法和手段

3.1 TCP 协议

作为 TCP/IP 协议簇中的骨干，TCP 协议基于“尽力而为”的网络层为应用层提供可靠的进程间通信服务，具体地说是可靠的全双工的端对端字节流传输服务。在 TCP 的协

议传输单元中 (TCP 报文段，TCP Segment)，收发双方以字节为单位使用序号 (Sequence Number) 明确收发的数据，使用 ACK 反馈 (Acknowledgment) 机制，实现端对端的可靠传输控制。

3.1.1 TCP 报文段 (Segment)

TCP 报文段结构如图1.3–1所示，采用 20 字节的报文段头以及不超过 40 字节的可选项。

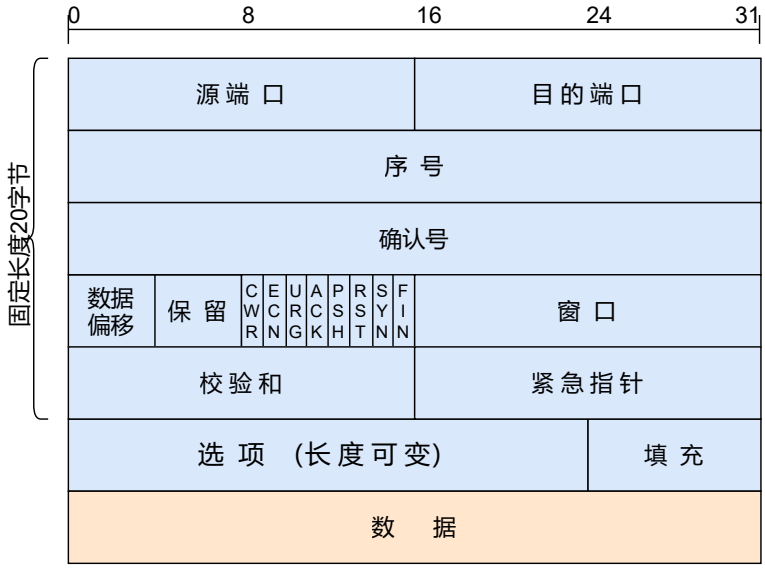

图 1.3–1 TCP 报文段结构示意图

主要字段如下：

1. **源端口号 (Source Port)**：16 位的源端口，与源 IP 地址一起标识发送该 TCP 报文段的通信进程。端口号范围 0-65535。
2. **目的端口号 (Destionation Port)**：16 位目的端口，与目的 IP 地址一起标识接收该 TCP 报文段的通信进程。
3. **序号 (Sequence Number)**：该 TCP 报文段中第一个数据字节的序号，占 4 个字节。在 TCP 连接建立时，通常生成一个随机数作为字节序号的初始值 (ISN)。
4. **确认号 (Acknowledgement Number)**：表示期望收到对方下一个报文段的字节序号，占 4 个字节。
5. **标志位 (TCP Flags)**：

(a) 确认 ACK(Acknowledgement)：置 1 表示确认号字段有效。

(b) 推送 PSH(Push)：置 1 表示该报文段优先级高，接收方 TCP 应尽快推送给接收应用程序。

(c) 复位 RST(Reset)：置 1 表示需要释放 TCP 连接并重新建立连接。一般称带 RST 标志的 TCP 报文段为“复位报文段”。

(d) 同步 SYN(Synchronization)：置 1 表示这是 TCP 请求连接报文段。一般称带 SYN 标志的 TCP 报文段为“同步报文段”。

(e) 终止 FIN(Finish)：置 1 表示发送方的数据已经发送完毕，并要求释放 TCP 连接。

6. **窗口大小 (Window)：**表示接收缓存大小。最早 TCP 协议首部只设置了 16 位的窗口大小，允许的最大缓存大小不超过 64KB；而 RFC1323 打破此限定，设置了 TCP 窗口缩放因子 (Window size scaling factor)，使窗口大小等于二者的乘积。

3.1.2 TCP 连接管理

为维护一个可靠的端对端传输，TCP 设计实现了完整的连接管理，重点是三次握手的连接建立和四次挥手的连接释放过程，如图1.3–2。

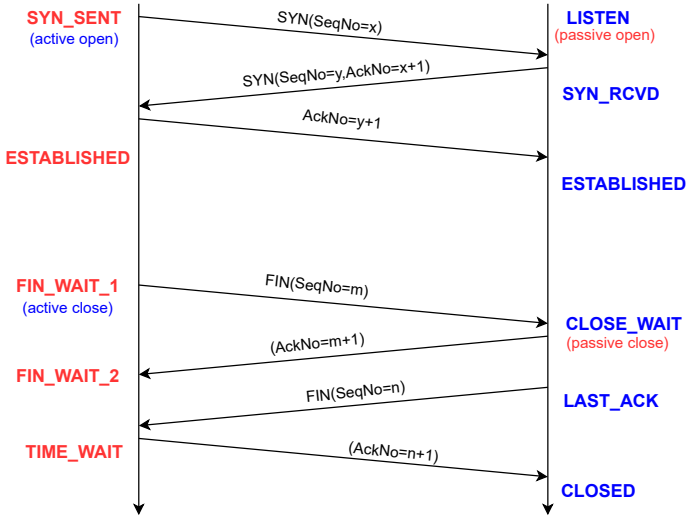

图 1.3–2 TCP 连接示意时序

建立过程：TCP 是面向连接的，数据传输之前必须在双方之间建立一条连接，并通过三次握手过程完成。其主要目的是，同步连接双方的序号和确认号，并交换 TCP 窗口大小等控制信息。一般地，客户端主动向服务器端发起连接请求，具体过程如下：

1. 第一次握手：建立连接。客户端发送连接请求报文段，将 SYN 位置为 1，Sequence Number 为 x；然后，客户端进入 SYN_SEND 状态，等待服务器的确认；

2. 第二次握手：服务器收到 SYN 报文段。服务器收到客户端的 SYN 报文段，需要对这个 SYN 报文段进行确认，设置确认号为 x+1；同时，将 SYN 位置为 1，序号为 y。服务器端将上述信息放入 SYN+ACK 报文段中，一起发送给客户端，此时服务器进入 SYN_RECV 状态；
3. 第三次握手：客户端收到服务器的 SYN+ACK 报文段之后，将确认号设置为 y+1，向服务器发送 ACK 报文段。这个报文段发送完毕以后，客户端和服务器端都进入 ESTABLISHED 状态，完成 TCP 三次握手。

释放连接 (四次挥手): 客户端没有新数据要发送时，就会释放 TCP 连接，发送一个报文段 (FIN=1，LEN=0)，进入 FIN-WAIT1 状态。服务器端收到后，返回客户端一个 ACK 报文段，进入 CLOSE-WAIT 状态。此时客户端进入 FIN-WAIT2 状态，不能再发送信息，但仍可接收信息。服务器端完成数据发送之后，也发出 FIN=1 的报文请求释放连接，并进入 LAST-ACK 状态，直至客户端返回确认（第四次挥手）才关闭 TCP 连接。客户端收到服务器的 FIN 报文后，则进入 TIME_WAIT 状态，并等待 2MSL(最大存活时间- Maximum Segment Lifetime) 时间之后，完全关闭 TCP 连接。(注：释放连接也可由服务器端先发起)

3.1.3 TCP 流量控制

流量控制的目的是让发送方的数据发送速率不要过快，以便接收方能及时接收。TCP 利用滑动窗口 (Sliding Window) 机制实施流量控制，其基本原理是用 TCP 报文段中的窗口大小字段来控制数据发送速率，即发送方的发送窗口应小于接收方回应报文中的通告窗口值。为了提高信道利用率，TCP 采用了连续 ARQ(Automatic Repeat reQuest)，TCP 两端都可以连续发出若干个分组然后等待确认，也都设有发送/接收窗口和发送/接收缓存。其中发送窗口可以用 3 个指针表示，而且发送窗口随着 TCP 报文段中窗口 (Window) 字段的数值动态变化。该窗口字段是告知对方自己还能接收多少字节的数据。发送方只发送序号在发送窗口之内的数据，避免发送过快，从而实现流量控制。具体的缓存、窗口和指针变化过程请参阅教材相关内容。

若接收方没有空余缓存，就会发送零窗口大小 (Window=0) 的报文段，即要求发送端停止数据发送。然而，当接收方释放出足够缓存后，发送方往往无法及时恢复数据发送，这就产生了死锁问题。为解决零窗口 (Zero-Window) 问题，TCP 为每一个连接设置一个持续计时器 (persistence timer)。只要 TCP 的一方收到对方的零窗口报文段，就启动该计时器，并周期性的发送一个长度为 1 字节的探测报文段 (周期亦会增加)。收到探测报文段的一方，在返回 ACK 报文段时，将使用 Window 字段发送其最新缓存大小；若最新缓存足够大，数据发送恢复，解决死锁问题。

3.1.4 TCP 拥塞控制

计算机网络中的带宽、节点 (路由器和主机) 中的缓存和处理机等，都属于网络的资源，在某段时间，若对网络中某一资源的需求超出该资源所能提供的可用部分，网络的性能就会变坏，这种情况就叫做拥塞。TCP 拥塞控制的目的是避免过多的数据注入网络引发路由器或链路资源过载。拥塞控制是一个全局性的系统化的工程，将报文段丢失视作网络拥塞发生的信号，通过调整拥塞控制窗口 (cwnd - Congestion Window) 和利用 ACK 反馈机制来控制数据发送速率。

TCP Tahoe 是 TCP 的最早版本，主要用三种算法去控制数据流和拥塞窗口：慢启动 (Slow Start)、拥塞避免 (Congestion Avoidance)、快重传 (Fast Retransmit)。TCP 事先并不知道网络资源的状况，首先采用慢启动算法开始发送数据：cwnd 初始值为 1 个 MSS(Maximum Segment Size)；每收到一个有效的 ACK 确认，cwnd+1，直至出现网络丢包超时或 cwnd 达到阀值 ssthresh（slow start thresh）。当 cwnd 等于或大于 ssthresh，采用拥塞避免算法，由 AIMD(加性增、积性减) 策略控制发送速率。若 TCP 报文段发送后 RTO (Retransmission Time Out) 时间仍未得到确认或收到三个确认号重复的 ACK 报文段，则启用快重传算法，并将 cwnd 重置为 1，ssthresh 减半。

TCP 拥塞控制算法一直处在不断的改进之中，围绕对网络环境因素感知和拥塞避免的控制，涌现出新的策略算法。TCP Reno 继承 Tahoe 的三个算法并增加了快速恢复 (Fast Recovery) 算法。收到三个重复的 ACK 后，Reno 会把当前的 ssthresh 的值设置为当前 cwnd 的一半，并将 cwnd 更新为 ssthresh+3MSS；然后每收到一个重复 ACK 则 cwnd+1，直至收到新确认号的 ACK 则将 cwnd 更新为 ssthresh。TCP NewReno 则进一步改进了快速恢复算法。

随着网络速度增长，传统拥塞控制算法的 cwnd 增长速度影响了 TCP 的性能，CUBIC[1] 应运而生。CUBIC 的关键特征是：cwnd 窗口的增长依赖两次丢包的时间。2016 年，谷歌提出了 BBR[2] 拥塞控制算法，它不再基于丢包感知来调整 cwnd，而是利用估算的带宽和延迟直接推测拥塞程度进而确定发送窗口。

3.2 实验方法和手段

两台实验机本地相互连接 (如图1.3–3)，在实验机中仿真不同的网络条件，以便观察 TCP 的各种控制现象。方案一使用 VMware Player 运行两台虚拟机，并通过“虚拟机设置-> 硬件-> 网络适配器-> 高级”（如图1.3–4）设置虚拟机的网卡传入/传出带宽、数据包丢失率、延迟等。方案二直接在两台 PC 机上操作，使用 tc 进行流量控制场景仿真，使用 wondershaper 对网卡进行限速。本实验需要使用的命令和工具，如表1.3–1所列。常用的 Linux 命令还包括：echo、cat、sysctl、ping、ftp 等。

[1] CUBIC: A New TCP-Friendly High-Speed TCP Variant

[2] BBR: Congestion-Based Congestion Control

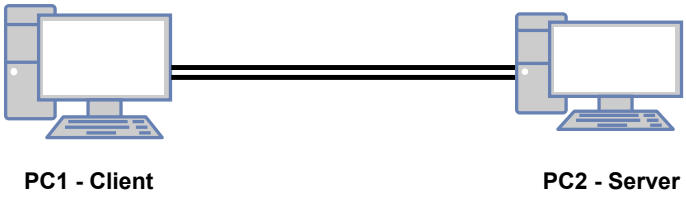

图 1.3–3 实验拓扑图

图 1.3–4 虚拟机网络适配器高级设置

四． 实验条件

1. 硬件：处于同一局域网的两台 PC 机 (可使用虚拟机也可使用物理机)。
2. 软件：Ubuntu 系统 (18.04 版)，预装 wireshark、curl、vsftp、netwox、telnet、nmap 和 iperf3。
3. 环境准备：分别以 PC1、PC2 作为 TCP 的客户端与服务端；启动两台实验机后，可使用 ping 进行连接性测试，也可使用 nmap 扫一下对方打开的端口，确保实验环境正常 (指导书中 PC2 的 IP 为 192.168.100.144，实验中应替换为实际的 IP) 。

参考资料：

- TCP 协议 RFC
- 计算机网络--自顶向下方法：Wireshark 实验指导书–TCP
- Congestion Control in Linux TCP

表 1.3–1 主要工具及命令列表

命令	作用	参考
ifconfig	配置网络	https://man.linuxde.net/ifconfig
nmap	网络扫描	https://nmap.org/man/zh/index.html
curl	文本浏览器	https://man.linuxde.net/curl
wget	下载 Web 文件	wget <IP>/<PathAndFileName>
tc	流量控制	tc 命令手册
iptables	防火墙配置	https://man.linuxde.net/iptables
netwox	网络工具	https://sourceforge.net/projects/ntwox/
ss	Socket 状态	ss -atn
netstat	显示网络状态	netstat -atn
wondershaper	网卡限速工具	wondershaper [网 口] [下 载 速 率] [上 行 速 率] wondershaper clear [网口]
iperf3	网络性能分析	https://iperf.fr/

五． 实验步骤

5.1 TCP 正常连接观察

1. 利用 python 自带的 SimpleHTTPServer 模块，在 PC2 上启动一个简易的 web 服务器。终端上运行 `echo "TCP lab test" > index.html` 创建 index.html 文件为测试站首页，运行 `sudo python -m SimpleHTTPServer 80` 启动一个简易 web 服务器；打开新终端，键入 `ss -tln` 查看当前主机打开的 TCP 连接，确认 80 端口处理监听状态。
2. 在 PC1 上打开一个终端，键入 `sudo wireshark` 启动抓包软件；再打开一个新终端，键入 curl <PC2 的 IP> ；停止抓包，在 wireshark 过滤出 TCP 类型报文。观察首个 TCP 报文头，并分析各段值代表的意义。如果想要关闭相对序号/确认号，可以选择 Wireshark 菜单栏中的 Edit→Preference→protocols→TCP，去掉 Relative sequence number 勾选项。使用 Wireshark 内置的绘制流功能，选择菜单栏中的 Statistics→Flow Graph，Flow Type 选择 TCP flows 可以直观地显示 TCP 序号和确认号是如何工作的。
3. 观察 TCP 三次握手与四次挥手报文，注意报文收发过程中，双方 TCP 状态的变化。以本次捕获的报文为依据，分别画出本次 TCP 连接三次握手与四次挥手的时序图，结合 TCP 状态机，在双方各阶段标出对应的 TCP 状态。选择其中一个 TCP 报文，配合 Wireshark 截图，分析该报文 TCP 首部各字段的定义、值及其含义。

5.2 异常传输观察分析

1. 尝试连接未存活的主机或对未监听端口。
   (a) 用 curl 访问一个不存在的主机 IP，抓包观察共发送了几次 SYN 报文。根据每次时间间隔变化，估算 RTO（重传超时）。
   (b) 查看 Linux 主机的系统的 TCP 参数 SYN 重传设定：
   `cat /proc/sys/net/ipv4/tcp_syn_retries`
   (c) 更改 SYN 重传次数为 3：
   `echo "3" > /proc/sys/net/ipv4/tcp_syn_retries`
   (d) 再次 curl 访问，观察抓包内容。
   (e) 关闭服务器端的 SimpleHTTPServer(ctrl+C 中断，或关闭所在终端)，客户端 curl 访问服务器 80 端口，观察应答报文。
   (f) 运行 `nmap -sS <PC2 的 IP>` 扫描服务器，并抓包。
   (g) 在报告中总结以上观察结果，解释 SYN 扫描原理。
2. 客户端发送了第一个 SYN 连接请求，服务器无响应的情景。
   (a) 服务器开启 telnet 或 ssh 服务，客户端先尝试连接服务器，连接成功后，在双方键入 ss -tan 查看所有 TCP 连接状态。我们看到的 TCP 连接建立过程同 1 中的 HTTP 访问类似。在客户端，利用 iptables 拦截服务器回应的 SYN ACK 包，命令如下：
   `sudo iptables -I INPUT -s 192.168.100.144 -p tcp -m tcp --tcp-flags ALL SYN, ACK -j DROP`
   (b) 再次尝试连接并启动 wireshark 抓包，并在双方多次用 ss -tan 观察 TCP 状态。
   (c) 观察 TCP 的状态变化，分析 wireshark 捕获的 TCP 异常报文。
   (d) 服务端的 SYN-RECV 状态何时释放?
   (e) SYN ACK 重传了几次，时间间隔有何变化?
   (f) 参考 1 中的操作，在服务端修改 SYN ACK 重传次数 (tcp_synack_retries)，再次观察，此任务结束后清空防火墙规则 (iptables -F)。
3. SYN 洪泛。在服务器端 `sudo echo "0">/proc/sys/net/ipv4/tcp_syncookies` 禁用 syncookies，通过 `sudo sysctl -w net.ipv4.tcp_max_syn_backlog = 6` 指定所能接受 SYN 同步包的最大客户端数量为 6；在客户端运用 netwox 工具对服务器监听的端口产生大量 SYN 连接请求 (如 `sudo netwox 76 -i 192.168.100.144 -p 23`)，再使用正常的连接工具 (如 telnet) 尝试连接，观察交互情况 (特别是服务器端的 TCP 连接状态)，抓包分析，解释 SYN 洪泛攻击原理与对策。
4. 异常报文分析。在服务器端产生一个 100M 的大文件，利用 1 中的 web 服务器，在客户端上

用 wget 下载它。参考命令：

(a) 产生一个 100M 文件：

```
dd if=/dev/zero of=100M.file bs=1M count=100
```

(b) 模拟网络延迟、包重复、包乱序、包损坏：

```
tc qdisc add dev ens33 root netem delay 70ms 10ms 30% duplicate 1% reorder 5%
10% corrupt 0.1%
```

(调整上述命令中的数值，以达到期望效果；将此行命令的 add 改为 change/del 即修改/删除此规则。)

(c) 下载服务器上的大文件：`wget 192.168.100.144/100M.file`

抓包记录以上过程，分析黑色标签错误报文，结合 TCP 实现机制，分析这些报文产生的原因。此类报文也可以从现实网络行为捕获，请结合捕获的报文进行分析，报文附件随报告提交。包括但不限于以下几种类型报文：[Duplicate ACK]、[TCP Retransmission]、[Fast Retransmission]、[TCP Spurious Retransmission]、[TCP Out-Of-Order]、[TCP Previous segment not captured]。

5.3 流量控制

1. 编写一对简单的 TCP 连接程序，也可以直接运行指导书提供的 Python 程序 (源代码见本节最后的附件)。在客户端快速发送数据给服务端，而服务端则有意缓慢地接收数据，观察 TCP 如何用窗口大小值进行流量控制。虚拟机两端分别运行 `python3 server.py` 和 `python3 client.py`。
2. 捕捉两端通信报文数据，分析报文中的 Win 值变化，联系上下报文，解释为什么出现 [TCP Windows Full]、[TCP ZeroWindows]、[TCP Keep-Alive] 等和窗口大小相关的流量控制报文。捕获的原始报文存成附件随实验报告提交。

5.4 拥塞控制

1. 任一端限制网卡传入/传出带宽为 10Mbps 以下：使用虚拟机作为实验机，可在 VMWare Player 中的虚拟机设置 → 网络适配器 → 高级中设置；物理机可使用 wondershaper 命令进行限速。再启动应用 (可以是 http wget，也可以 ftp 下载/上传) 传输大文件观察。
2. Wireshark 捕捉全部传输过程数据，找出该网络活动的拥塞点，并结合 Analyze→Expert Information、Statistic→I/O Graphs、Statistic→TCP Stream Graphs(如图1.3–5)，分析此传输过程中的慢启动、拥塞避免、快速恢复等阶段。
3. TCP 竞争观察：类似以上试验，我们在一个大文件传输过程中，迅速启动另一应用，双向进行大文件传输，观察两路（或两路以上）TCP 连接的速率变化。

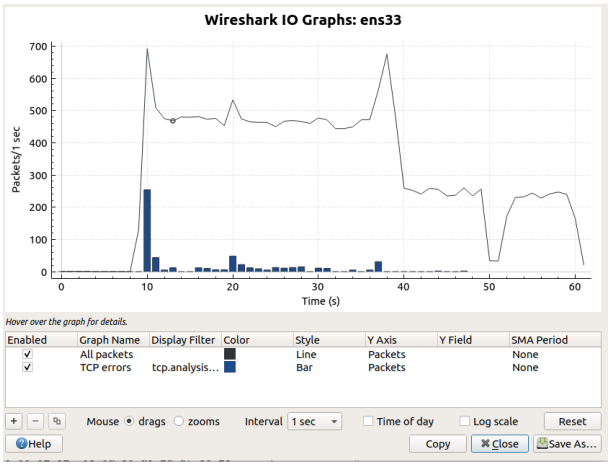

图 1.3–5 I/O Graphs

4. TCP 拥塞控制算法比较：运用 iperf3 性能分析工具，设置 ubuntu 系统的 tcp_congestion_control 策略，分析比较不同网络环境下、不同拥塞控制算法的表现。参考如下命令（灵活组合使用）：
   (a) 可用策略：
   ```
   cat /proc/sys/net/ipv4/tcp_available_congestion_control
   ```
   (b) 修改策略：
   ```
   sysctl -w net.ipv4.tcp_congestion_control=bbr
   ```
   (c) iperf3 服务端启动：`iperf3 -s -p 5555`
   (d) iperf3 客户端启动：`iperf3 -c 192.168.100.144 -p 5555`
   (e) 设置链路带宽、时延、丢包率可以使用虚拟网卡高级属性配置，也可以参考前面使用 tc 命令配置。

六． 思考题

1. TCP 在不可靠的 IP 层上建立了可靠的端对端连接，如果要在不可靠的 UDP 上建立可靠的端对端传输系统，需要考虑哪些方面？
2. TCP 连接建立过程中，存在哪些等待队列？这些队列是否可能出现溢出状况？该如何避免？
3. 本次实验观察了 Linux 环境下的 TCP 实现，在 Windows、macOS 环境下，操作系统又是如何实现 TCP 的呢？类似 Linux TCP 参数，在不同系统环境下如何查看或设置，请尝试通过捕捉其通信过程、比较其实现异同。
4. 在 TCP 状态机 (图1.3–6) 中，有些状态停留时间较长，易观察到，有些状态很短暂不易观察到。试列出不易观察到的状态，并考虑观察到它们的可能方法。
5. TCP 是封装单元为 MSS，可是我们在捕捉过程中常发现远大于此值的 TCP 报文，为什么 TCP

可以提交如此大的报文呢？此类型的包远超出链路层的 MTU，它是如何被处理的呢？请从两端同时抓包观察比对。

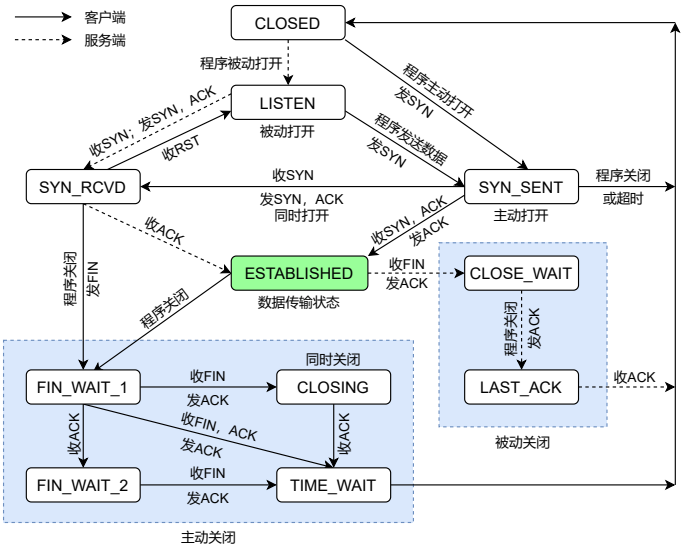

图 1.3–6 TCP 连接管理状态机

七． 注意事项及有关说明

1. Linux 上运行 wireshark 抓包需要 root 权限，请先在终端中通过 su 命令切换为 root 后，再键入 `wireshark &` 启动；或是打开个终端键入 `sudo wireshark`。`netwox/netwag` 同样需要 root 权限。
2. Ubuntu 中装有 python2 和 python3 两种类型版本，实验中用了 python2 启动简易 Web 服务器，附件中的程序用 Python3 运行。
3. 环境还原：前面操作的 iptables、tc 遗留规则可能会影响后面的操作效果，`iptables --list` 查看核对一下当前的规则，`iptables -F` 清空当前规则；同样，使用 `tc qdisc del dev eth0 root RULE` 清除网卡 eth0 队列规则；wondershaper 限速使用 clear 参数清除。使用虚拟机的快照功能是更原始、更彻底的还原方式。
4. 批量网络扫描是危害网络行为，仅在实验室环境下进行试验学习，不得用于运营网络。

八． 考核方法

完成本次实验，并提交一份实验报告和一组 Wireshark 数据存储文件。报告内容应当包括以下部分，相关的分析解释都对应有截图证明，并与数据存储文件吻合。

1. (20 分) 正确绘制出了三次握手报文与四次挥手报文 (须结合捕获的报文)，并正确标识出了各阶段 TCP 状态；
2. (20 分) 观察传输异常现象，并进行分析；
3. (20 分) 完成流量控制操作要求，结合上下分析报文窗口变化，解释说明相关的类型报文成因。
4. (20 分) 完成拥塞控制操作要求，成功从捕获的数据分析出一次完整 TCP 流的各个阶段（慢启动、拥塞控制、快速恢复）。
5. (10 分) 完成任 2 道思考题。
6. (10 分) 记录自己在本次实验中所遇到的问题，以及心得感悟。如果遇到异常情况，或者无法完成任务时，也请分析错误产生的原因。

九． 附件

为了方便进行实验，提供一份 Python3 套接字通信作为附件，具体代码如下：

- 服务端 server.py

```python
import socket
import time
def recv_data(sock,length):
    data=b''
    while len(data) < length:
        more = sock.recv(length - len(data))
        if not more:
            pass
        data += more
    return data
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 设置接收缓冲区大小为 1024
s.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 1024)
s.bind(('0.0.0.0', 9999))
s.listen(1)
print("Sever listening on 9999...")
try:
    sc, addr = s.accept()
    while True:
```

```
        rcvdata = recv_data(sc,16)
        time.sleep(0.01)
        pass
finally:
    sc.close()
    s.close()
```

- 客户端 client.py

```
import socket
import time
ip_port = ("192.168.100.144", 9999)
data = "0123456789\n"*100
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    c.connect(ip_port)
    print("Connected.")
    time.sleep(1)
    i=0
    while i<10:
        print("Send:"+data)
        c.send(data.encode())
        i=i+1
finally:
    print("Sent. Waiting....")
while True:
    time.sleep(0.001)
```