2023－2024学年第1学期

计算机网络课程设计（Ⅰ）

题目： 基于WinPcap/Npcap的网络协议分析器/编辑器

班级： 21034101

学号： 211540882

姓名： 董建明

教师： 胡泽

成绩：

项目	程序（50%）	报告（50%）
分数		

目　录

1 系统概述

1.1 选题

设计开发一个基于 Scapy 的协议分析器，协议分析器能实现捕获、分析数据包的功能。

1.2 系统实现功能

基础工作：MAC 协议、ARP 协议、IP 协议、TCP/UDP 协议的分析。

额外工作：IPv6 协议、ICMP 协议、DNS 协议的分析；界面添加单选组件。

1.3 背景知识

Scapy：Scapy 是一个交互式数据包处理的 Python 库，它允许用户发送、嗅探、分析和伪造数据包。这种能力允许构建能够探测、扫描或攻击网络的工具。

Tkinter：Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序。

Threading：Python 通过两个标准库 thread 和 threading 提供对线程的支持。thread 提供了低级别的、原始的线程以及一个简单的锁。threading 模块在低层级的 _thread 模块之上构造了高层级的线程接口。

Datetime：datetime 模块提供用于处理日期和时间的类。在支持日期时间数学运算的同时，实现的关注点更着重于如何能够更有效地解析其属性用于格式化输出和数据操作。

MAC 协议：多路访问控制协议(multiple access control protocol)，采用分布式算法决定结点如何共享信道，即决策结点何时可以传输数据，必须基于信道本身，通信信道共享协调信息。

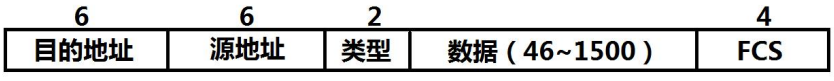

图 1 MAC 帧格式

ARP 协议：地址解析协议（Address Resolution Protocol），是根据 IP 地址获取物理地址的一个 TCP/IP 协议。在 IP 以太网中，当一个上层协议要发送数据包时，将包含目标 IP 地址的 ARP 请求广播到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址。

图 2 ARP 报文格式

IP 协议：互联网协议（Internet Protocol），是 TCP/IP 协议栈中最核心的协议之一，通过 IP 地址，保证了联网设备的唯一性，实现了网络通信的面向无连接和不可靠的传输功能。

图 3 IP 报文格式

TCP 协议：传输控制协议（Transmission Control Protocol）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。TCP 旨在适应支持多网络应用的分层协议层次结构。 连接到不同但互连的计算机通信网络的主计算机中的成对进程之间依靠 TCP 提供可靠的通信服务。TCP 假设它可以从较低级别的协议获得简单的，可能不可靠的数据报服务。

图 4 TCP 报文格式

UDP 协议：用户数据报协议（User Datagram Protocol）。UDP 为应用程序提供了一种无需建立连接就可以发送封装的 IP 数据包的方法。 UDP 是一个简单的面向消息的传输层协议，尽管 UDP 提供标头和有效负载的完整性验证（通过校验和），但它不保证向上层协议提供消息传递，并且 UDP 层在发送后不会保留 UDP 消息的状态。因此，UDP 有时被称为不可靠的数据报协议。如果需要传输可靠性，则必须在用户应用程序中实现。

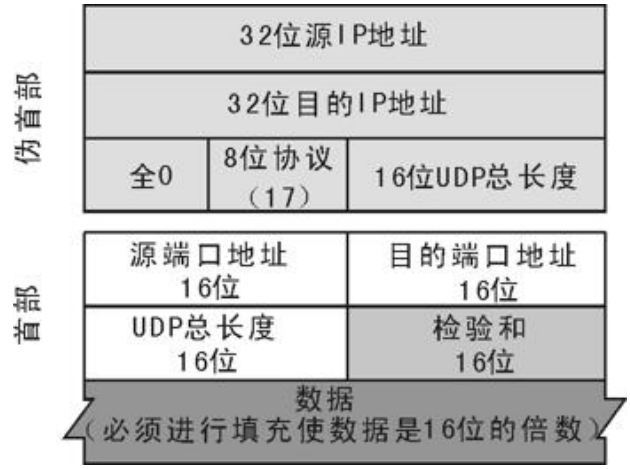

图 5 UDP 报文格式

IPv6 协议：互联网协议第 6 版（Internet Protocol Version 6），相比于 IPv4 协议，IPv6 协议有更大的地址空间、扩展的地址层次结构、灵活的首部格式、改进的选项、改进的选项、允许协议继续扩充、支持即插即用（即自动配置）、支持资源的预分配、最小的 MTU 变为 1280 字节。

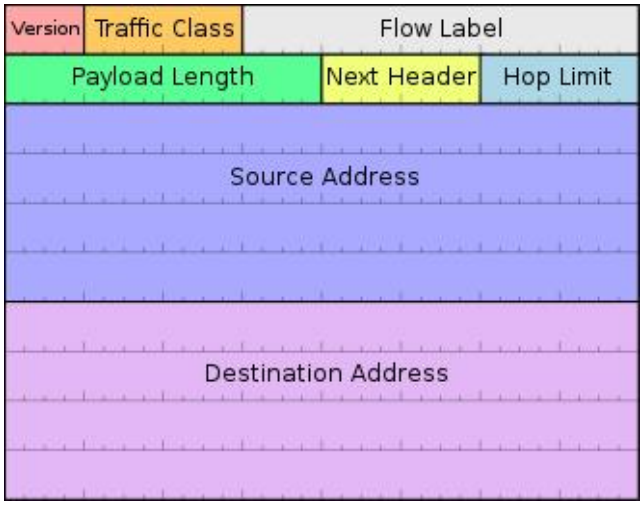

图 6 IPv6 报文格式

ICMP 协议：互联网控制消息协议（Internet Control Message Protocol ），是网络层协议。它是 TCP/IP 协议簇的一个子协议，并不承载数据，也不是用来传输数据的。ICMP 是用来传递控制消息的，也就是我们经常说的：网络通不通，主机是否可达。常用的 ping 命令就是基于 ICMP。

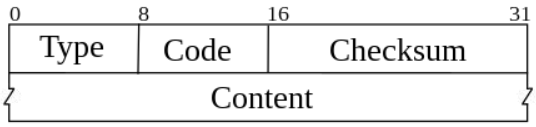

图 7 ICMP 报文格式

DNS 协议：域名解析协议（Domain Name System），将域名和 IP 地址相互映射，将域名映射成 IP 地址称为正向解析，将 IP 地址映射成域名称为反向解析。DNS 协议可以使用 UDP 或者 TCP 进行传输，使用的端口号都为 53。但大多数情况下 DNS 都使用 UDP 进行传输。

图 8 DNS 报文格式

2 开发环境

2.1 采用的操作系统

Windows 10 家庭版

2.2 集成开发环境

Pycharm 2023.1.3 专业版

Python 3.10.6

2.3 开发库的名称和版本号

表 1 开发库的名称和版本号

名称	版本号
Scapy 库	2.5.0
Tkinter 库	8.6 (内置库)
datetime 库	内置库
threading 库	内置库

3 系统设计

根据 analysisUI-接收-精简版-代码框架，为方便系统设计，将系统分为 6 个模块：项目结构如图 9 所示。

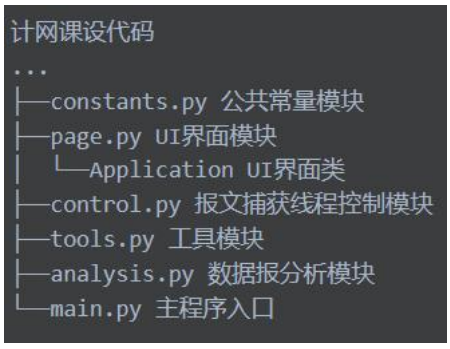

图 9 项目结构

4 程序流程

对于各模块中 8 各主要的报文分析函数，其流程图如下：

4.1 MAC 报文分析函数

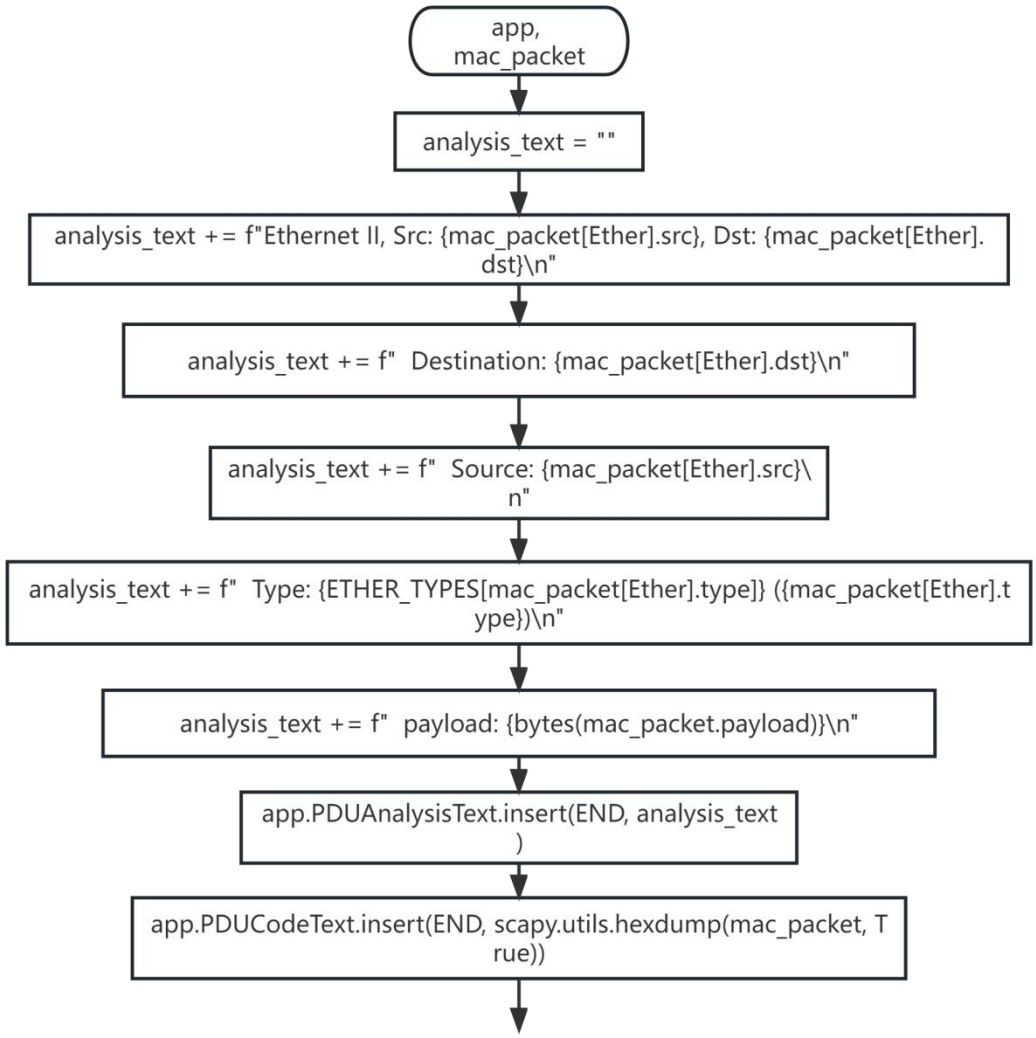

图 10 ether_pdu_analysis(app, mac_packet)

4.2 ARP 报文分析函数

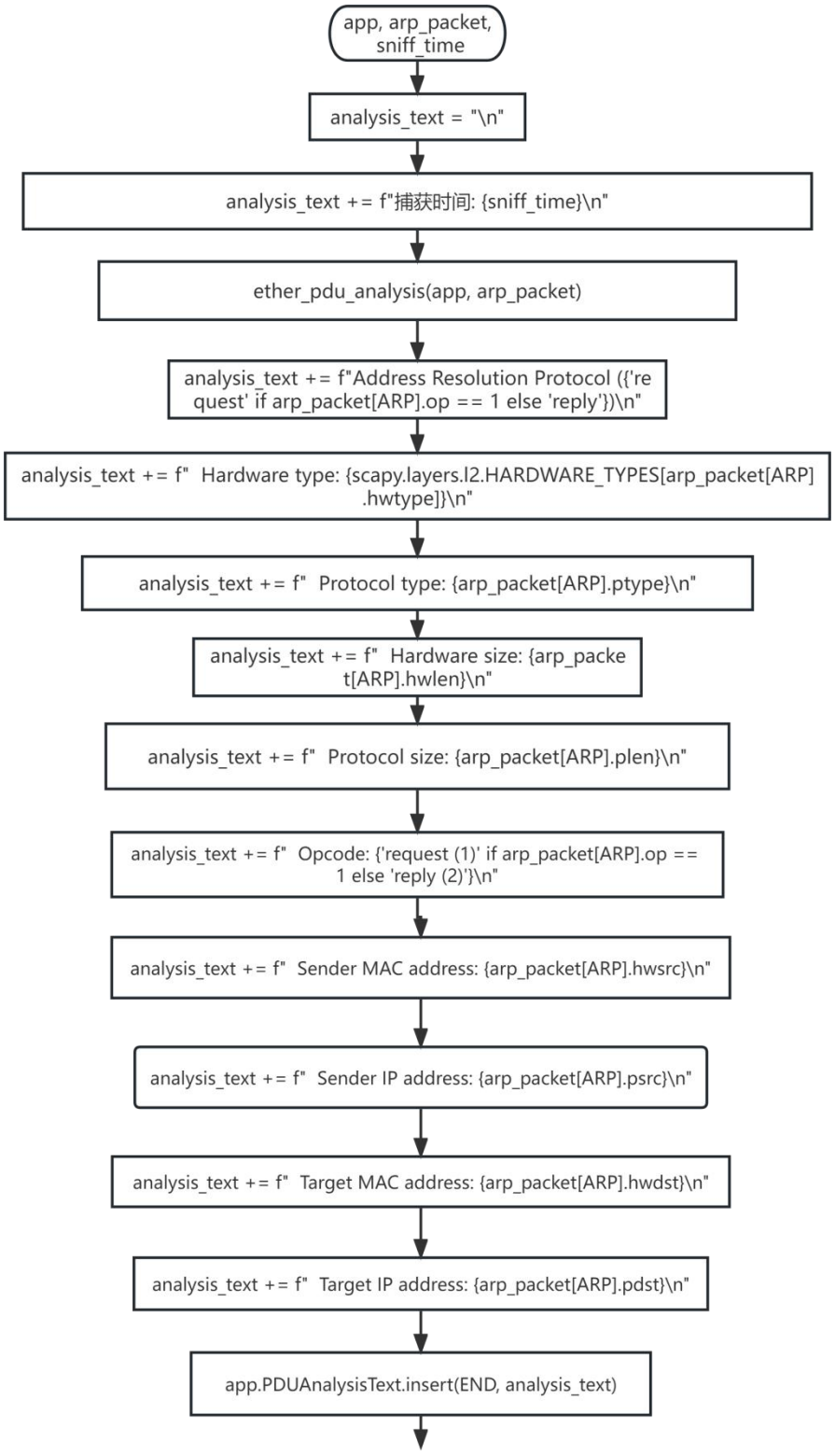

图 11 arp_pdu_analysis(app, arp_packet, sniff_time)

4.3 IP 报文分析函数

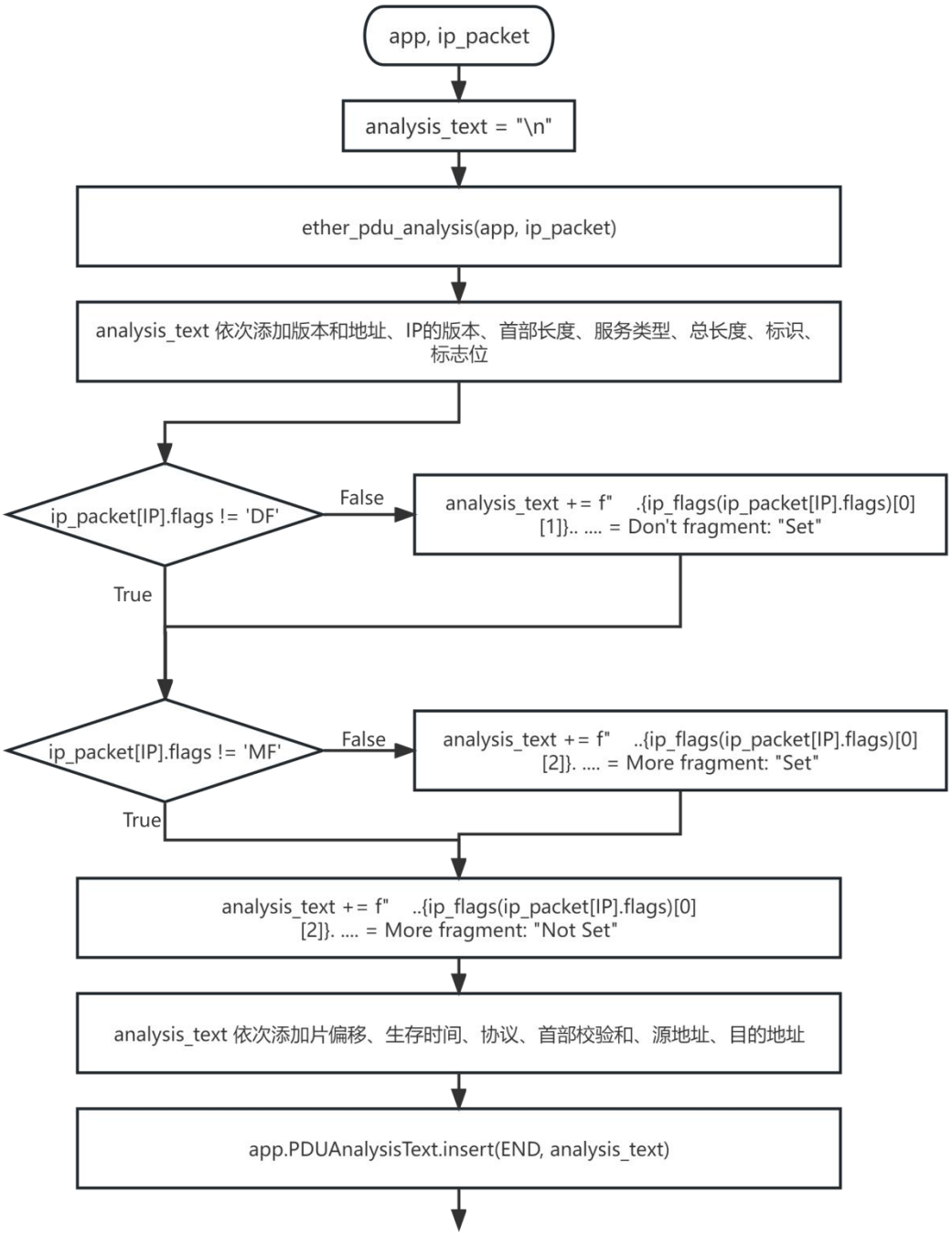

图 12 ip_pdu_analysis(app, ip_packet)

4.4 IPv6 报文分析函数

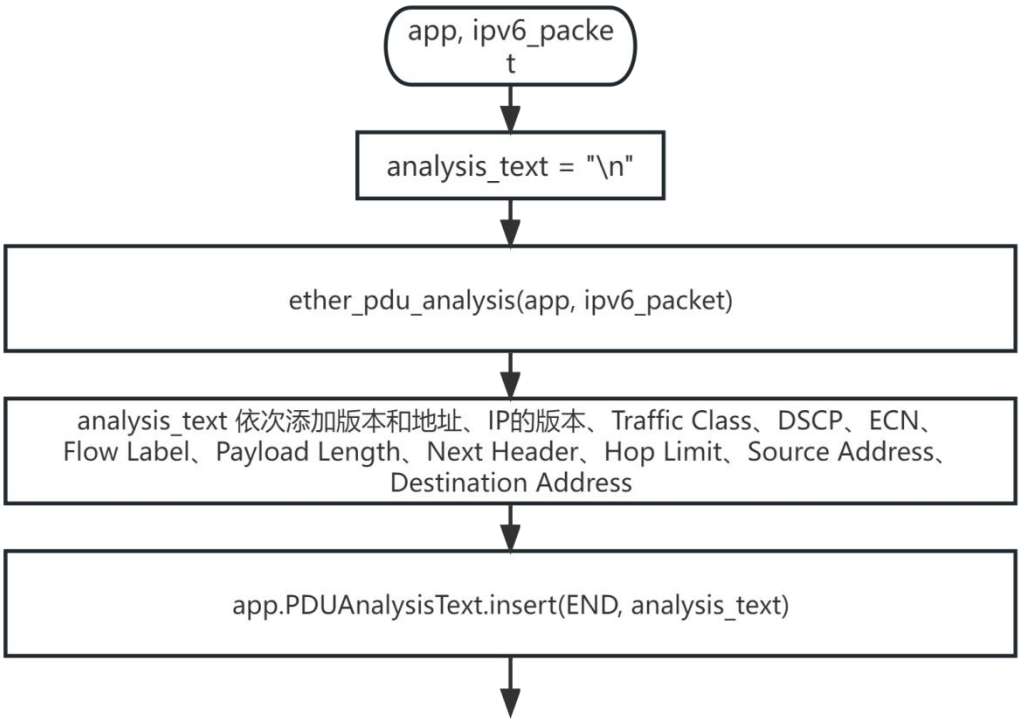

图 13 ipv6_pdu_analysis(app, ipv6_packet)

4.5 TCP 报文分析函数

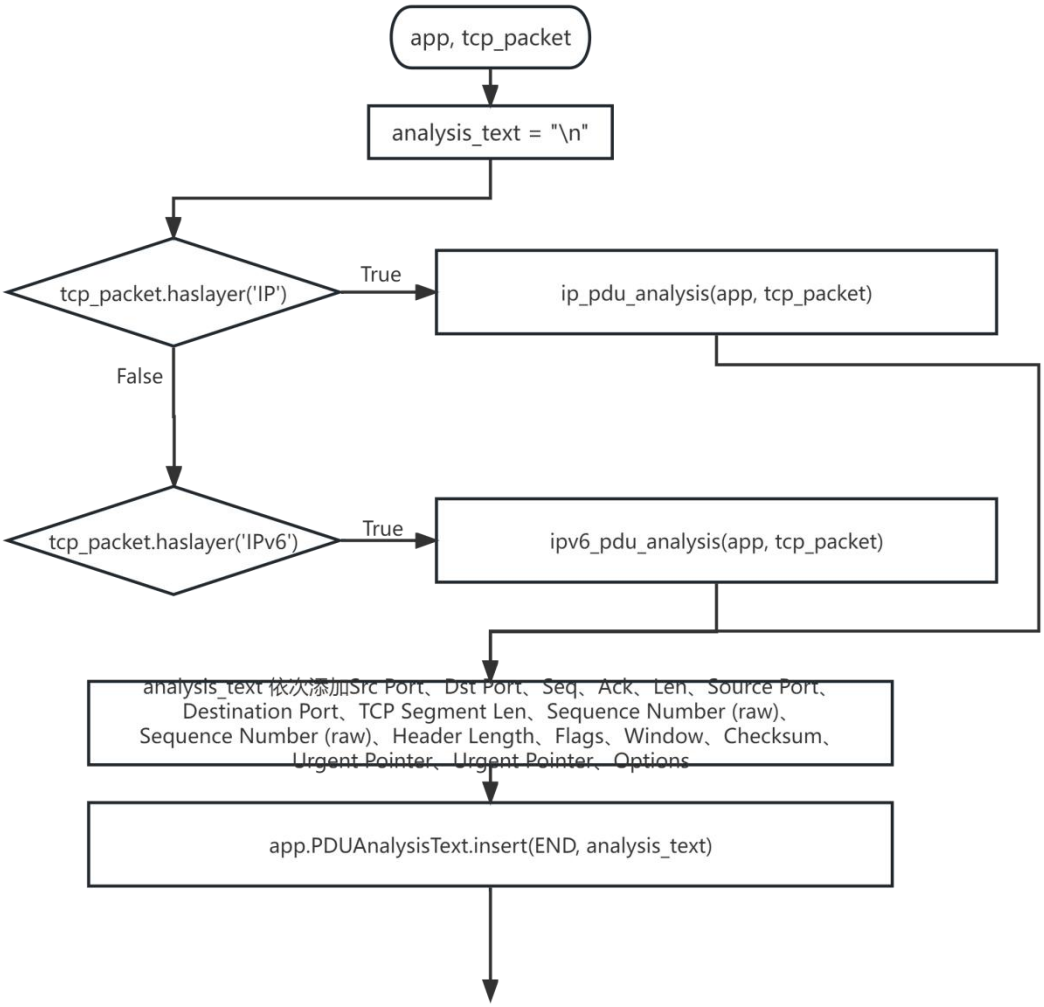

图 14 tcp_pdu_analysis(app, tcp_packet)

4.6 UDP 报文分析函数

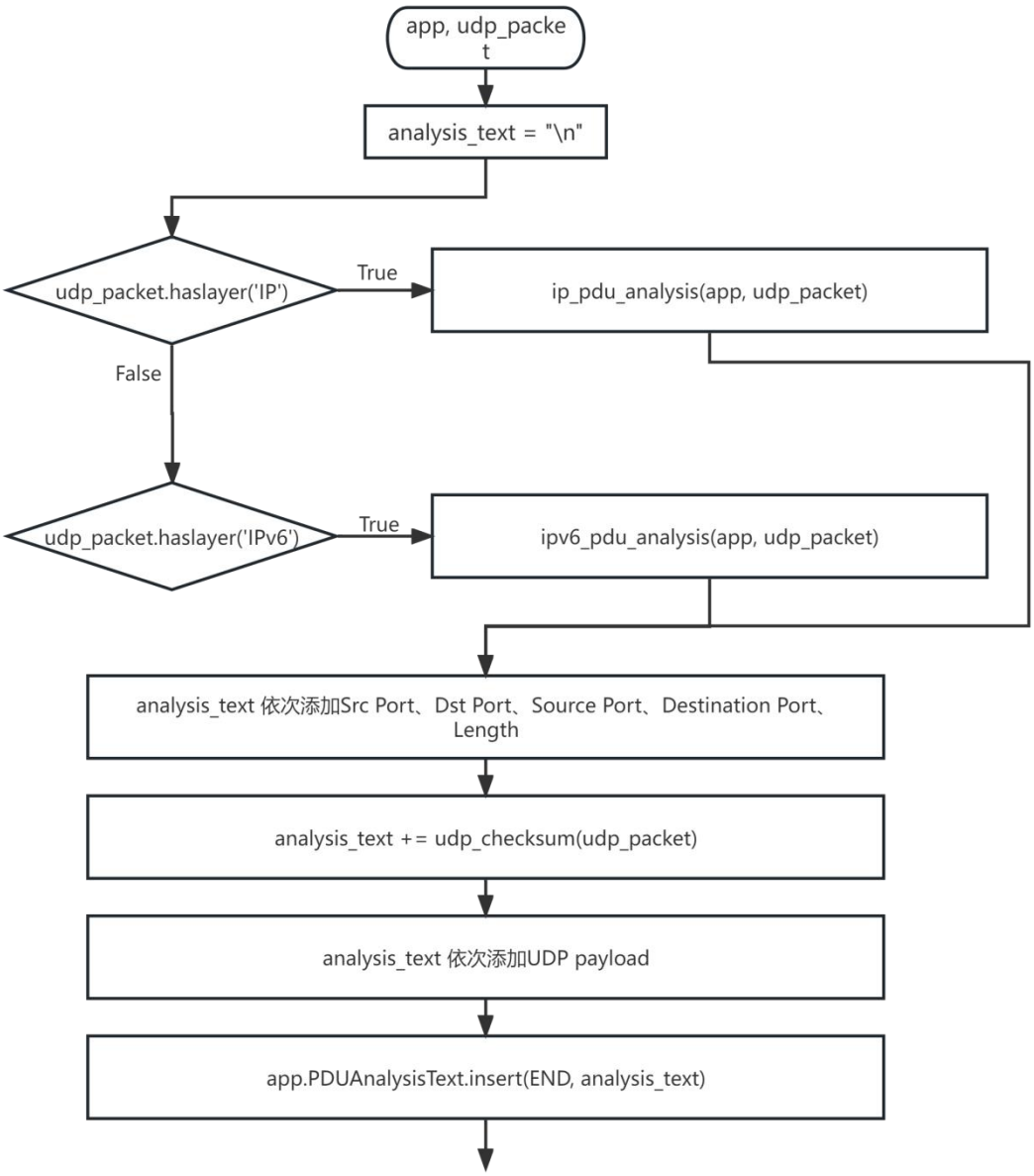

图 15 udp_pdu_analysis(app, udp_packet)

4.7 ICMP 报文分析函数

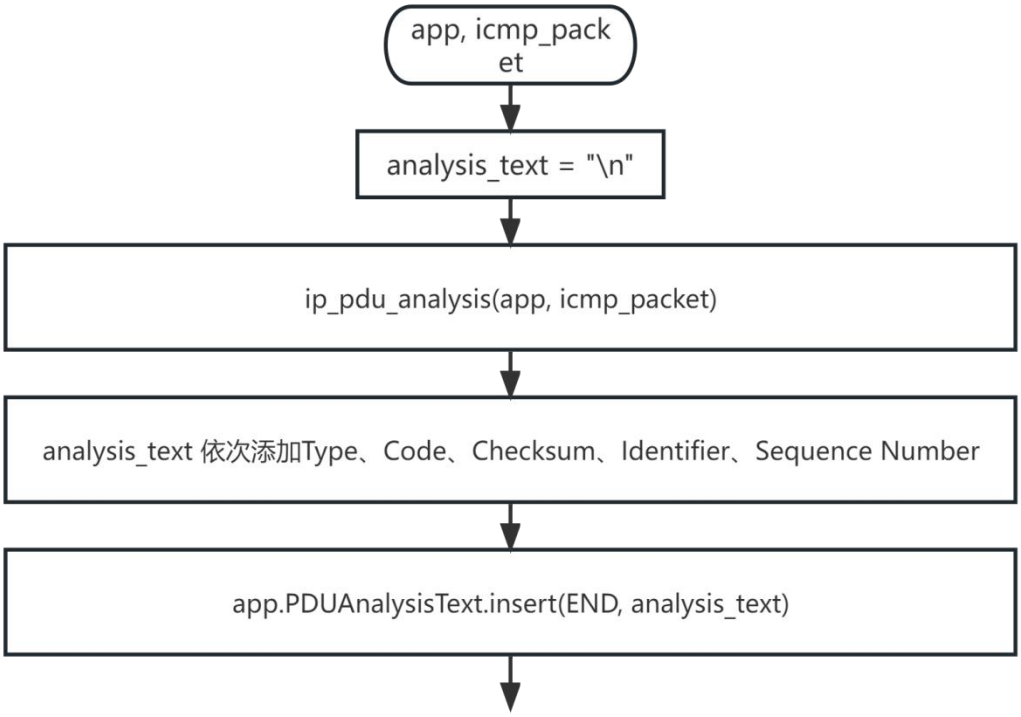

图 16 icmp_pdu_analysis(app, icmp_packet)

4.8 DNS 报文分析函数

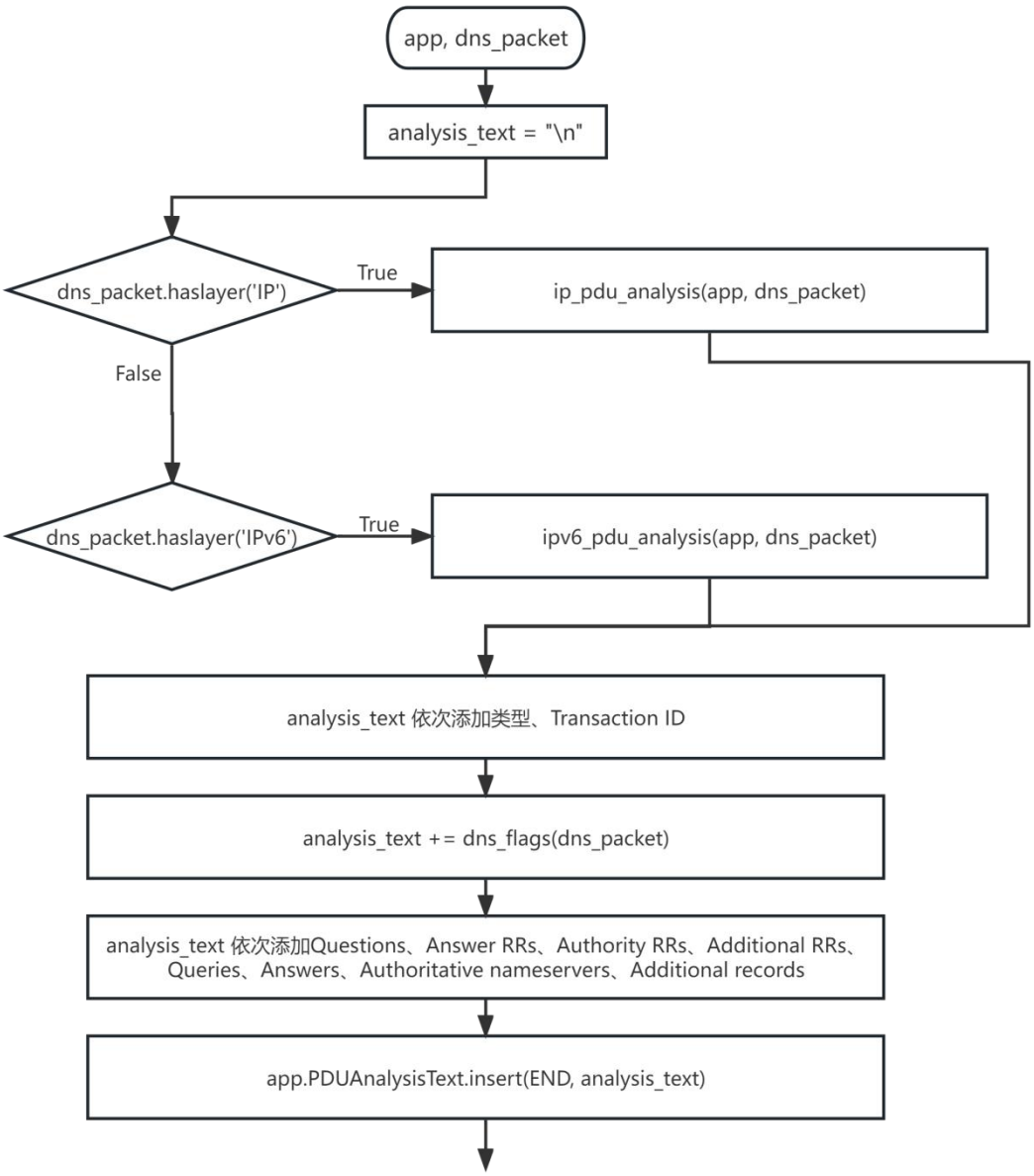

图 17 dns_pdu_analysis(app, dns_packet)

5 主要数据结构

创建 Application 类的对象时，初始化的数据及其数据结构如下：

表 2 主要数据结构

数据	类型	含义
self.count = None	Int	输入的待捕获

		数据帧数
self.countAct = None	Int	实际捕获的数据帧数
self.mainPDUShowWindow = None	PanedWindow 对象	协议分析主面板
self.countInput = None	Entry 对象	待捕获数据帧数输入框
self.conditionInput = None	Entry 对象	捕获条件输入框
self.startListenButton = None	Button 对象	开始捕获按钮
self.stopListenButton = None	Button 对象	停止捕获按钮
self.listbox = None	Listbox 对象	列表框显示捕获报文的摘要
self.PDUAnalysisText = None	Text 对象	显示报文分析文本组件
self.PDUCodeText = None	Text 对象	显示报文原始编码文本组件
self.sniffFlag = True	Boolean	捕获线程标志位
self.sniffDataList = []	List 其中各元素为 scapy.packet.Packet 类型	捕获的报文组成的列表
self.sniff_times = []	List 其中各元素为 datetime 对象	各报文捕获时间组成的列表

6 主要函数说明

表 3 主要函数

函数声明	功能	所作工作
createWidgets(self)	创建分析器界面	未修改
createControlWidgets(self)	创建控制面板	修改：1、pack()布局修改成 grid()布

		局方便添加单选组件 2、创建一组单选按钮
_sel(self, selected_option)	单选调用函数	新增
_quick_selection(self, control_frame)	创建单选按钮组，设置不同的值和文本标签　将单选按钮排列在一行中	新增
createPDUSumPanedWindow(self)	创建显示捕获报文的摘要的窗口	未修改
createPDUAnalysisPanedWindow(self)	创建显示捕获报文分层解析的窗口	未修改
createPDUCodePanedWindow(self)	创建显示捕获报文原始编码信息的窗口	未修改
chosen_pdu_analysis(self, event)	对选择的报文，判断其协议调用不同的分析函数 双击报文调用函数	修改：1、将捕获时间传递给 ARP 报文分析函数 2、清空解析信息和编码信息，以显示双击的报文信息 3、添加 IPv6、ICMP、DNS 协议的判断，判断其协议调用不同的分析函数
ether_pdu_analysis(app, mac_packet)	MAC 数据报分析	补全
arp_pdu_analysis(app, arp_packet, sniff_time)	ARP 数据报分析	补全
ip_pdu_analysis(app, ip_packet)	IP 数据报分析	补全
tcp_pdu_analysis(app, tcp_packet)	TCP 数据报分析	补全
udp_pdu_analysis(app, udp_packet)	UDP 数据报分析	补全
ipv6_pdu_analysis(app, ipv6_packet)	IPv6 数据报分析	新增

icmp_pdu_analysis(app, icmp_packet)	ICMP 数据报分析	新增
dns_pdu_analysis(app, dns_packet)	DNS 数据报分析	新增
start_sniff(app)	启动捕获线程	未修改
stop_sniff(app)	停止捕获线程	未修改
pdu_sniff(app)	捕获线程，捕获数据报，并调用回调函数	未修改
ip_monitor_callback(app, pkt)	回调函数，根据筛选条件调用不同的分析函数	修改：1、对每条报文的捕获时间进行记录 2、添加对 IPv6、ICMP、DNS 的判断，根据筛选条件调用不同的分析函数
int_bin(n, count, is_split=False)	10 进制转 2 进制，n：输入的 10 进制，count：输出的 2 进制位数，is_split：可选，是否四位一隔	修改：添加可选项 is_split：是否四位一隔
swap_endianness(n)	大端字节序转为小端字节序	新增
clear_data(app)	清空捕获数据	未修改
split_condition(app)	分割条件的函数 按空格	未修改
split_dul_equal(dul)	分割条件的函数，按==	为修改
ip_flags(chosen_ip_flags)	获取 ip 报文的标志位的 2 进制、16 进制、标志位信息	新增
ip_head_checksum(ip_packet)	IP 头部校验和 计算和验证	修改：判断头部校验和，返回值修改为直接返回要显示

		的字符串
tcp_flags(chosen_tcp_flags)	获取 TCP 的 Flag 每一位的值	修改：返回值修改为字典，包含标志位的 2 进制、16 进制、要显示的字符串、每个标志位代表的含义
pseudo_head(tcp_or_udp)	获取 TCP/UDP 伪首部 部分数据	新增
tcp_len(tcp_packet)	获取 TCP 长度	新增
tcp_checksum(tcp_packet)	获取 TCP 校验和信息	新增
udp_checksum(udp_packet)	获取 UDP 校验和信息	新增
icmp_checksum(icmp_packet)	获取 ICMP 校验和信息	新增
dns_flags(dns_packet)	获取 DNS 标志位信息	新增

7 系统使用说明

7.1 MAC 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 MAC 报文分析结果，报文原始编码信息窗口出现报文原始编码。

图 18 MAC 报文的捕获与解析

7.2 ARP 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 ARP 报文分析结果，报文原始编码信息窗口出现报文原始编码。

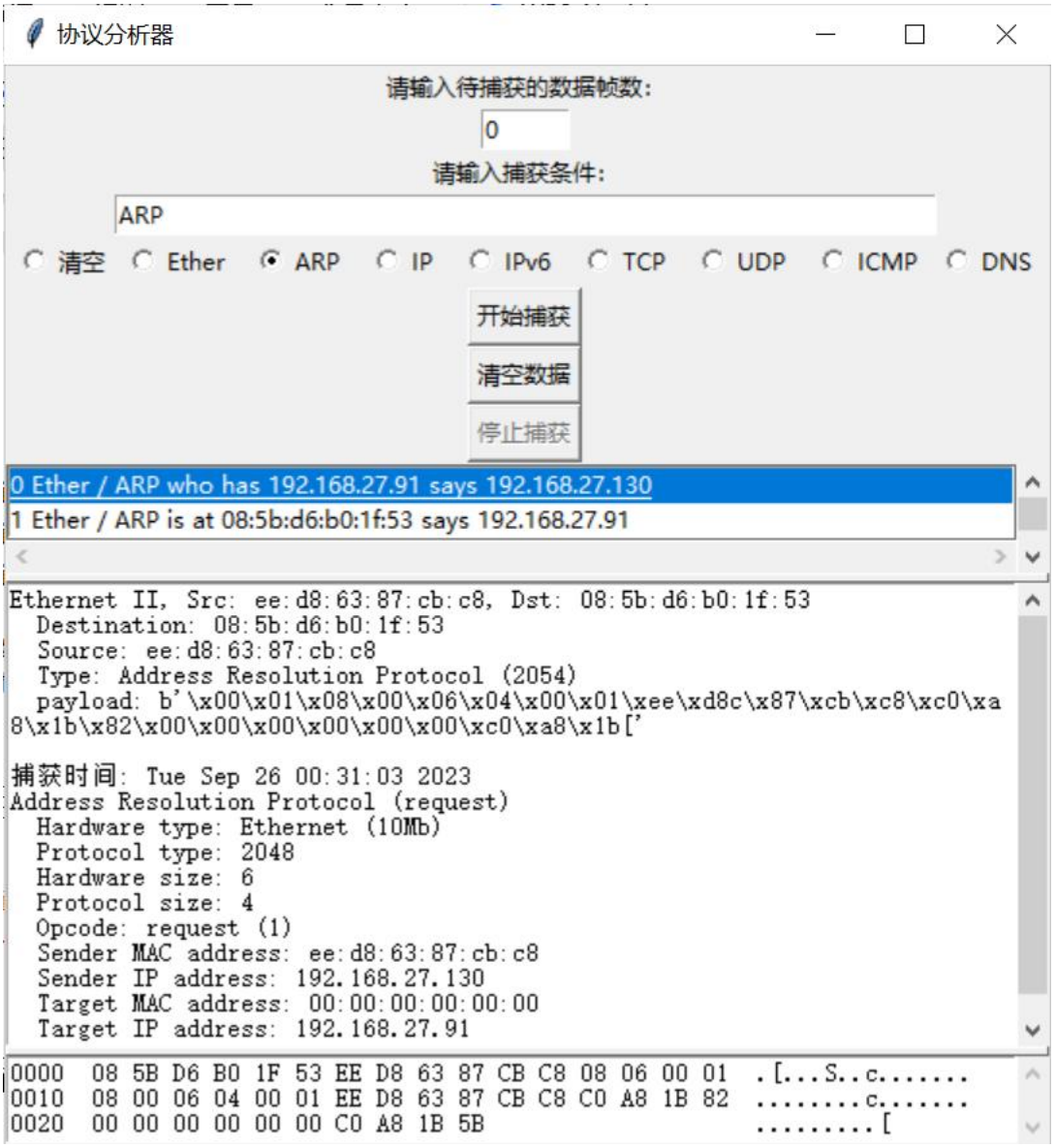

图 19 ARP 报文的捕获与解析

7.3 IP 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 IP 报文分析结果，报文原始编码信息窗口出现报文原始编码。

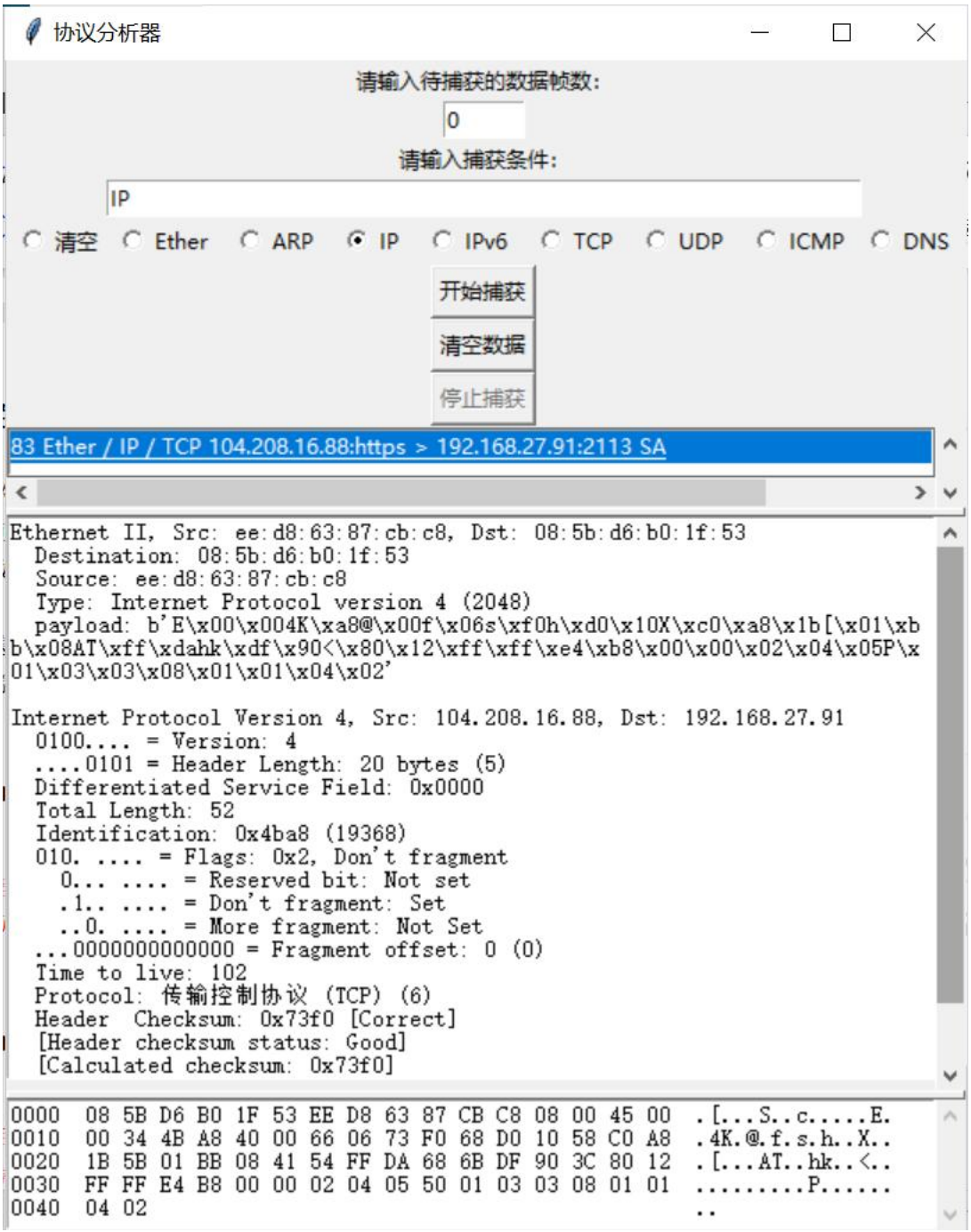

图 20 IP 报文捕获与解析（上部分）

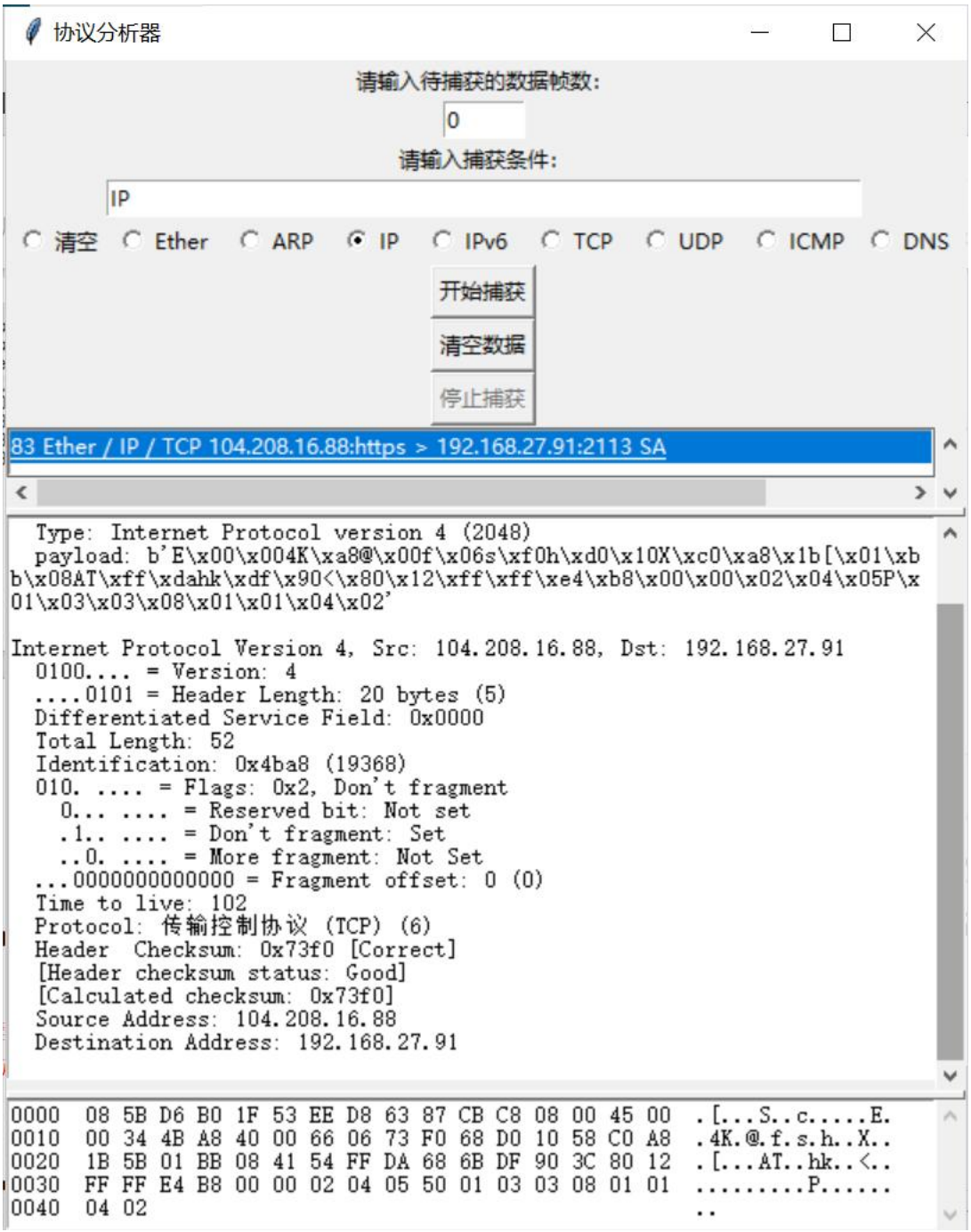

图 21 IP 报文捕获与解析（下部分）

7.4 IPv6 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 IPv6 报文分析结果，报文原始编码信息窗口出现报文原始编码。

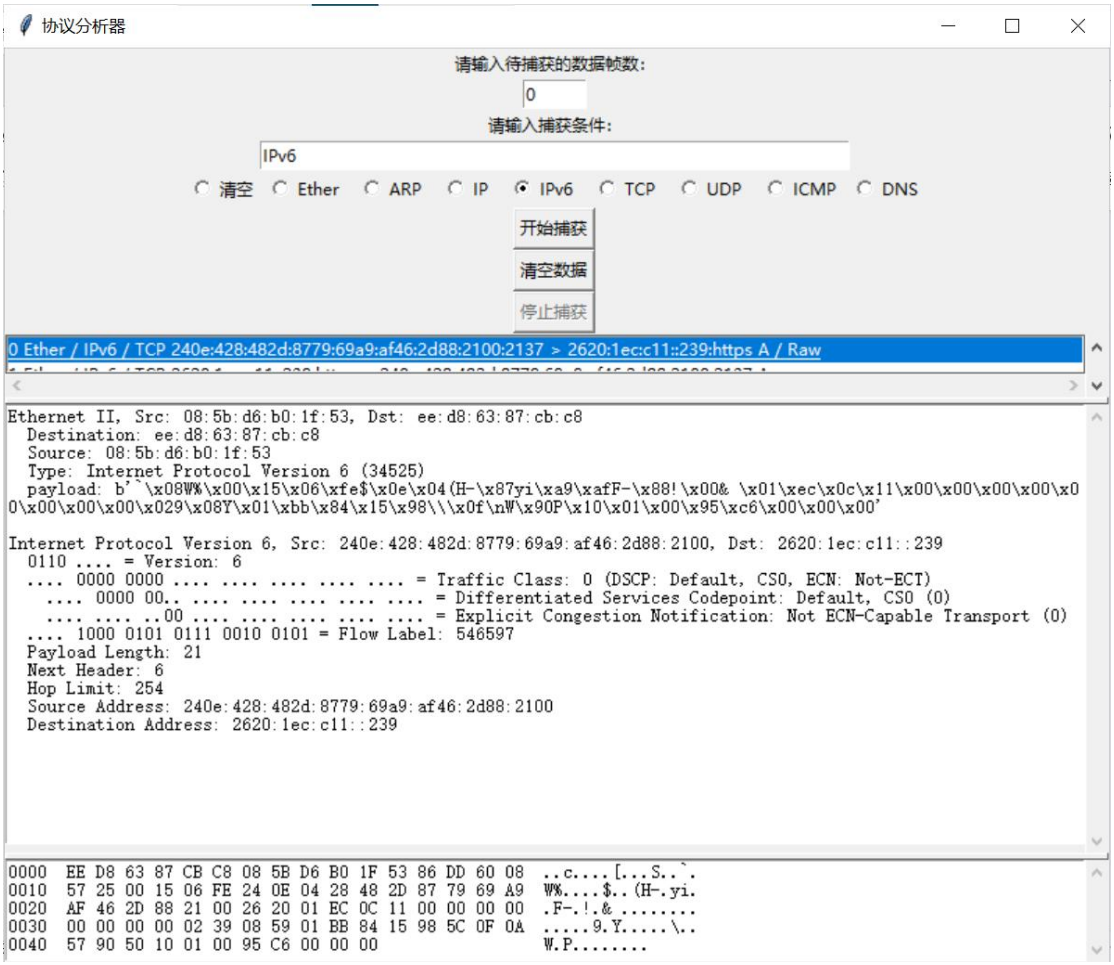

图 22 IPv6 报文捕获与解析

7.5 TCP 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 TCP 报文分析结果，报文原始编码信息窗口出现报文原始编码。

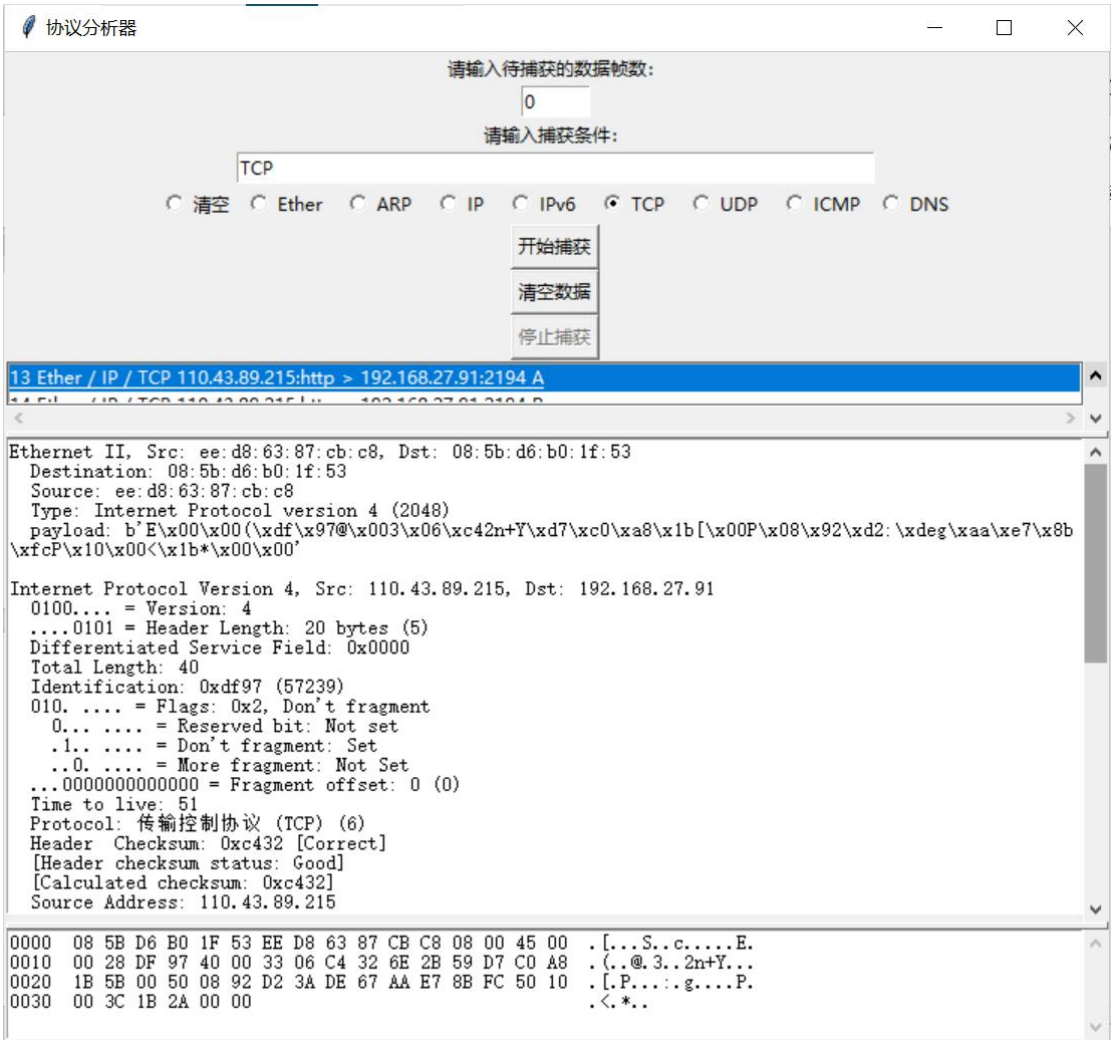

图 23 TCP 报文捕获与解析（上部分）

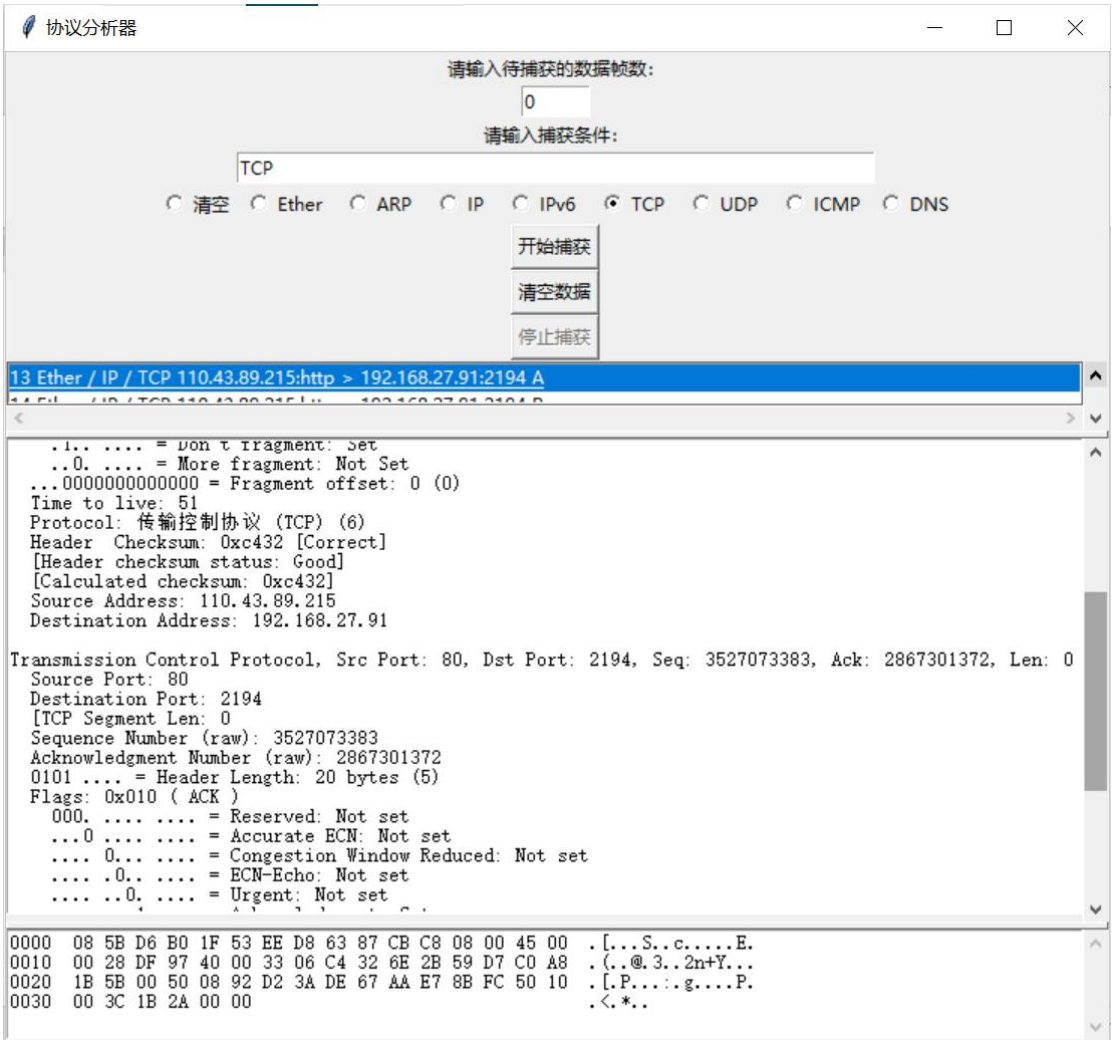

图 24 TCP 报文捕获与解析（中部分）

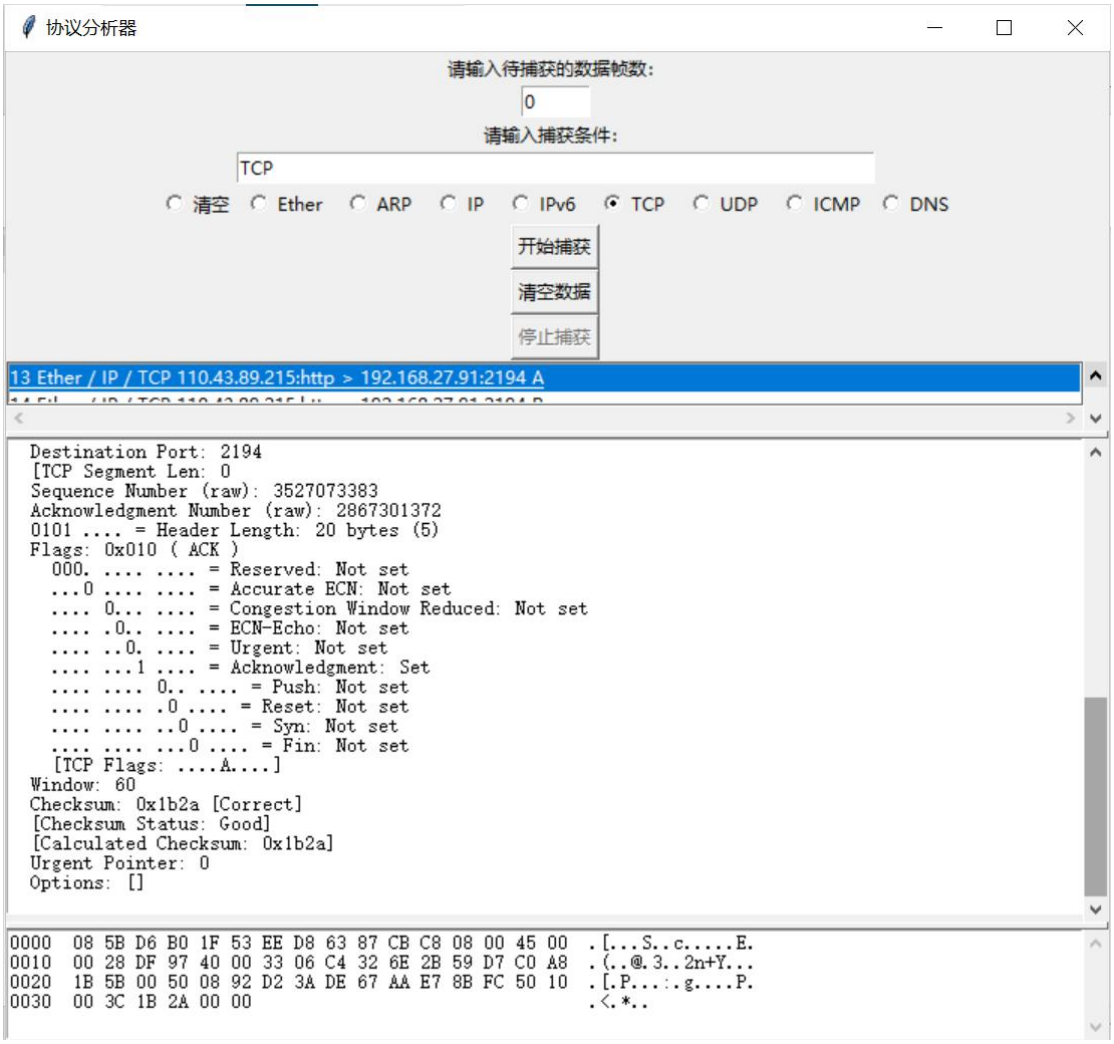

图 25 TCP 报文的捕获与解析（下部分）

7.6 UDP 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 UDP 报文分析结果，报文原始编码信息窗口出现报文原始编码。

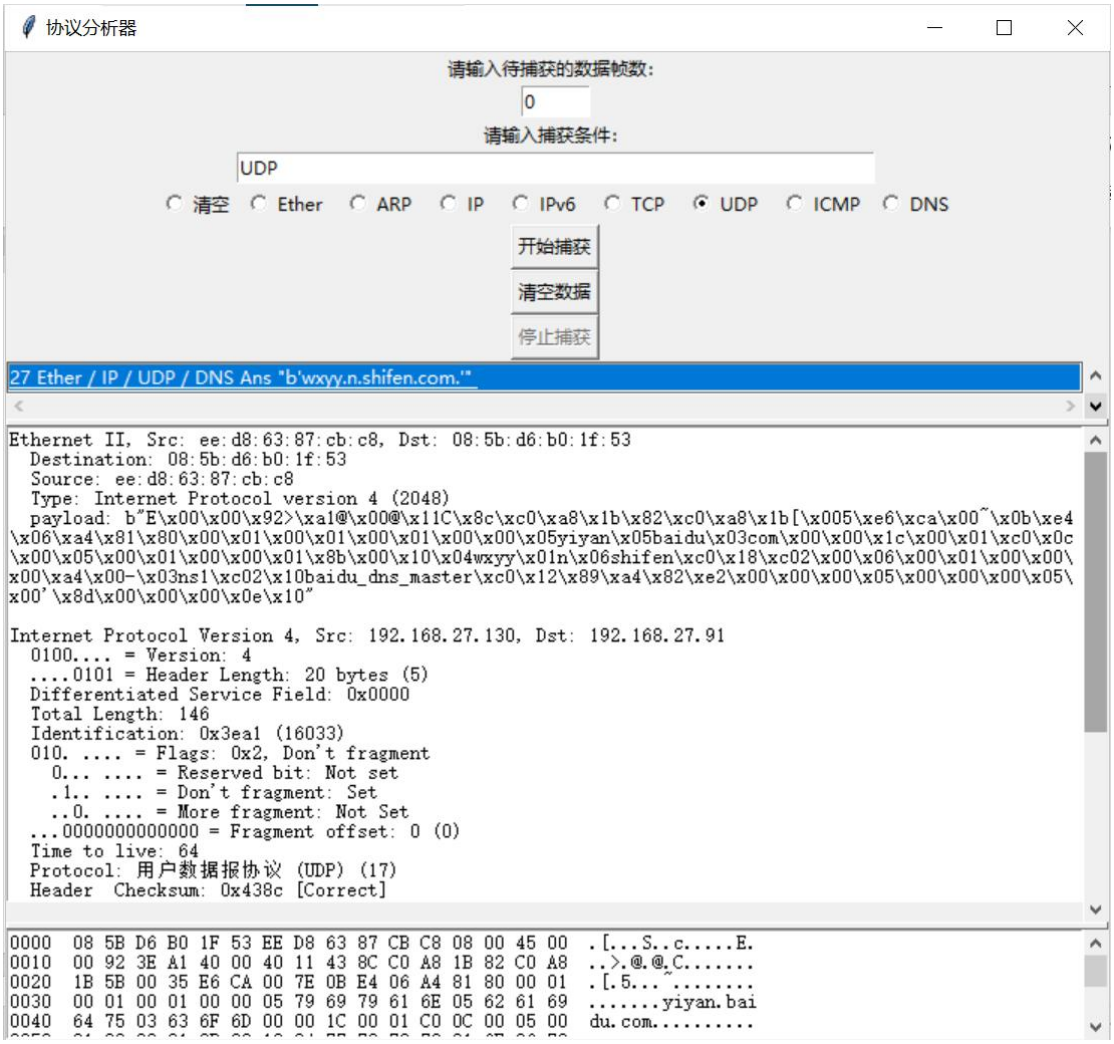

图 26 UDP 报文捕获与解析（上部分）

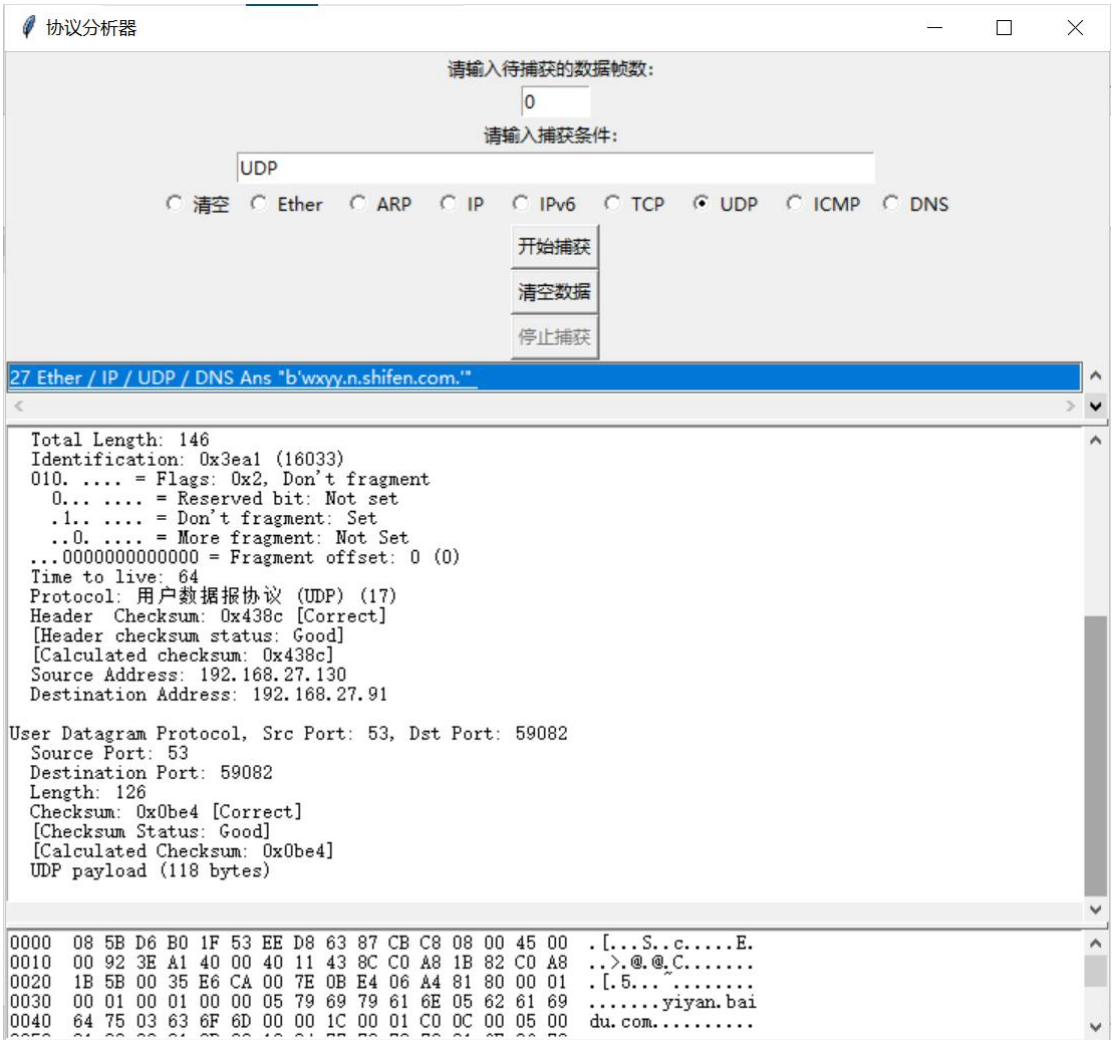

图 27 UDP 报文捕获与解析（下部分）

7.7 ICMP 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 ICMP 报文分析结果，报文原始编码信息窗口出现报文原始编码。

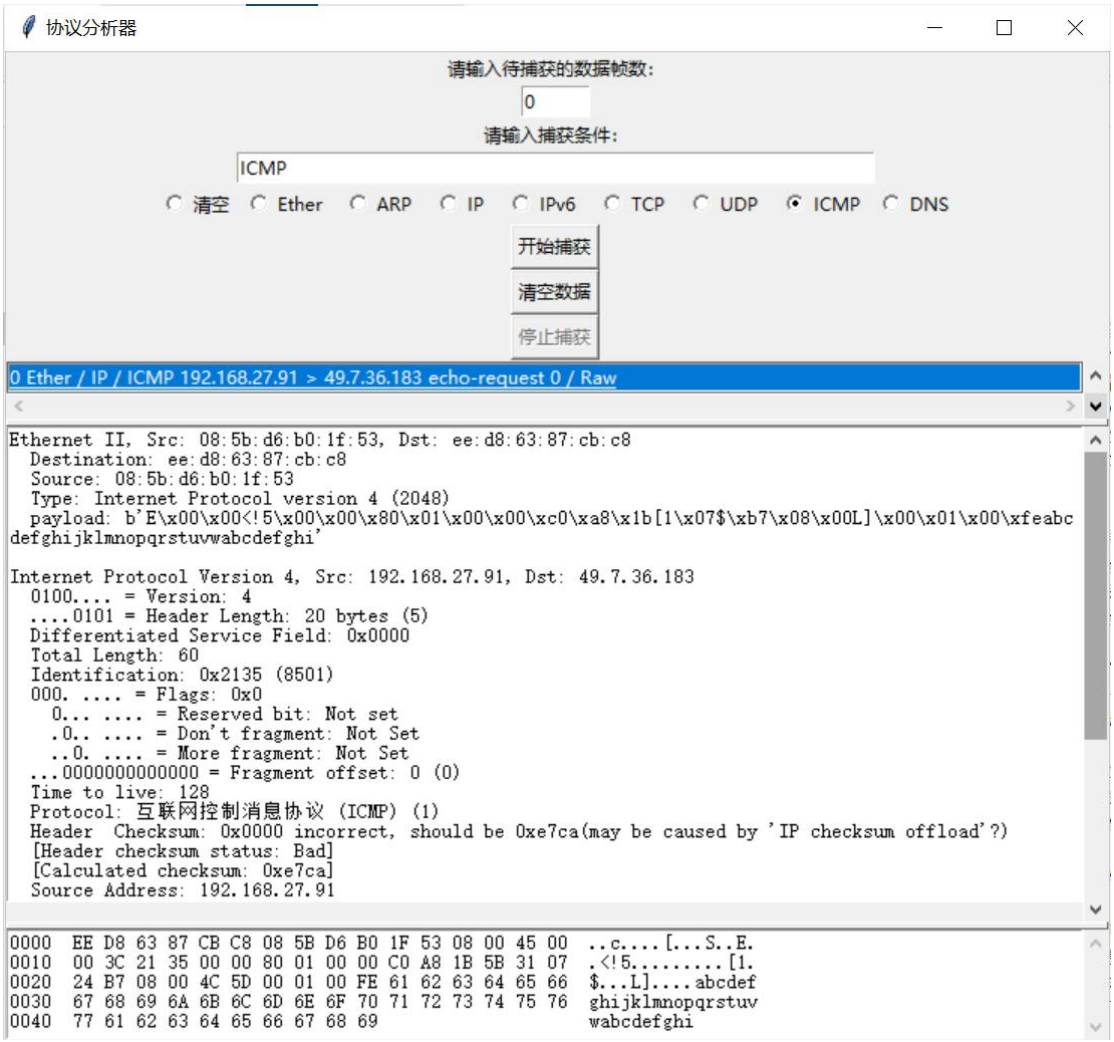

图 28 ICMP 报文捕获与解析（上部分）

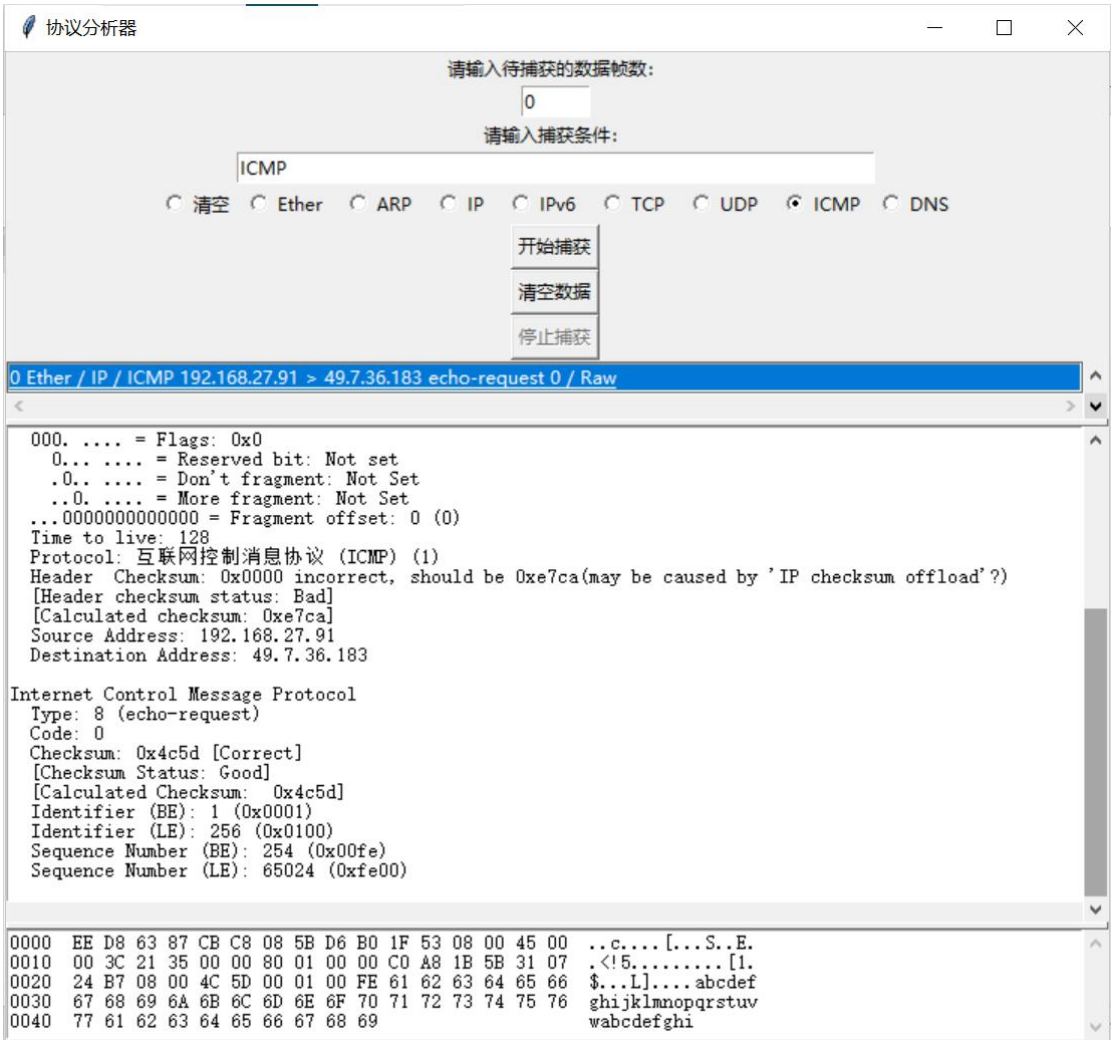

图 29 ICMP 报文捕获与解析（下部分）

7.8 DNS 报文的捕获与解析

输入捕获数据帧的个数，个数为 0 则一直捕获。

输入捕获条件，可以手动输入，也可以通过单选按钮快捷输入。

开始捕获，得到数据帧后，可以手动停止捕获，也可以等待其达到数目自动停止捕获。

双击报文简要信息，报文分层解析窗口出现 DNS 报文分析结果，报文原始编码信息窗口出现报文原始编码。

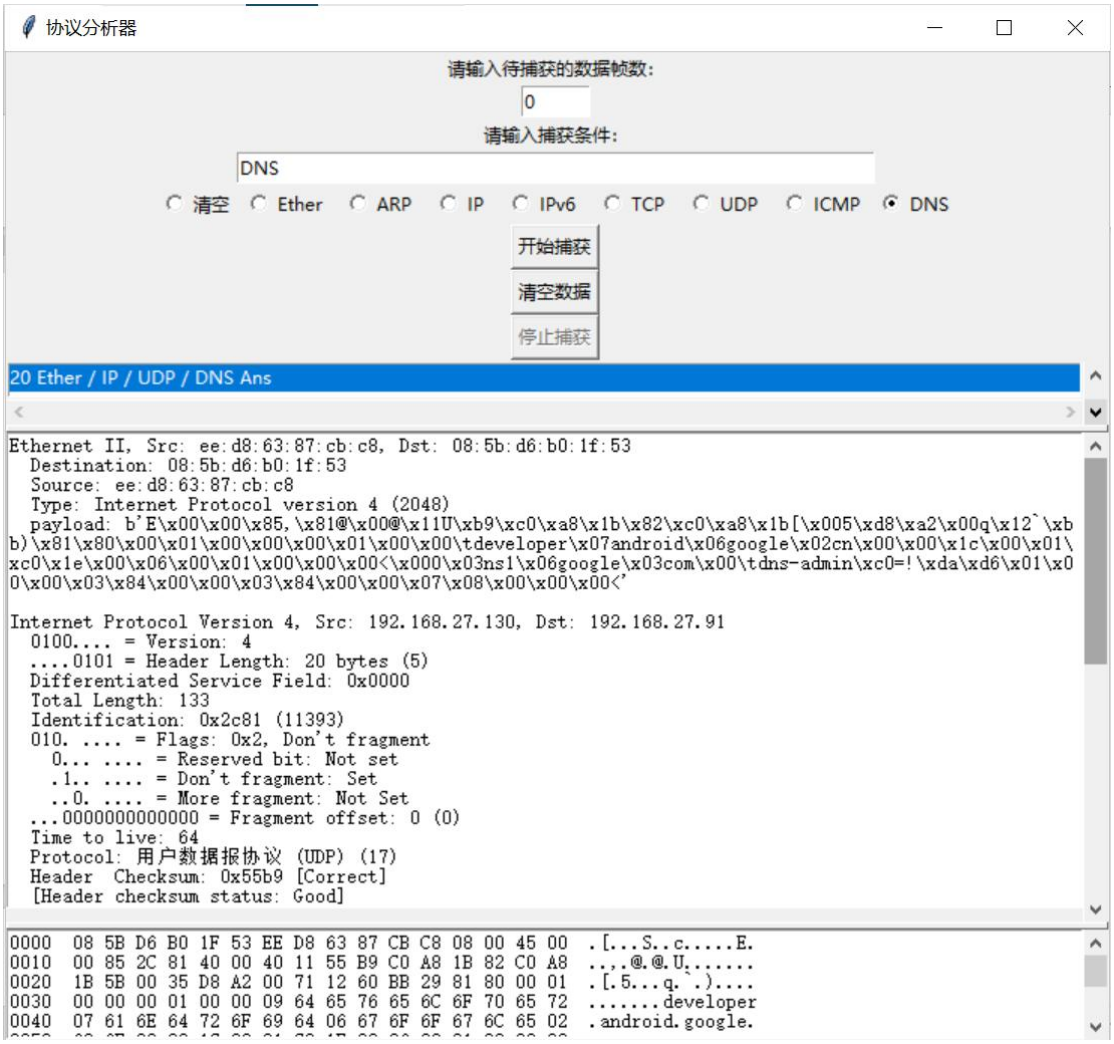

图 30 DNS 报文捕获与解析（上部分）

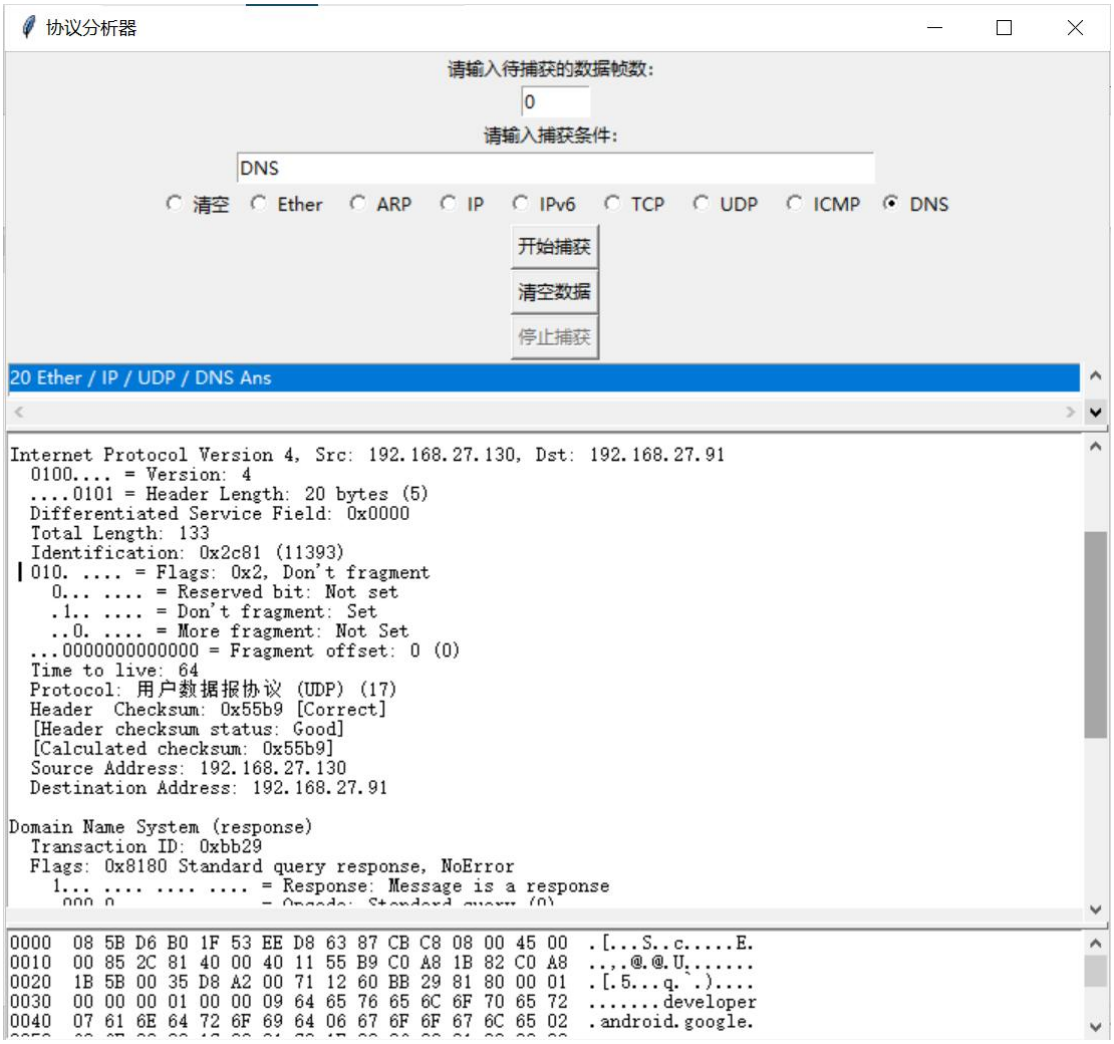

图 31 DNS 报文捕获与解析（中部分）

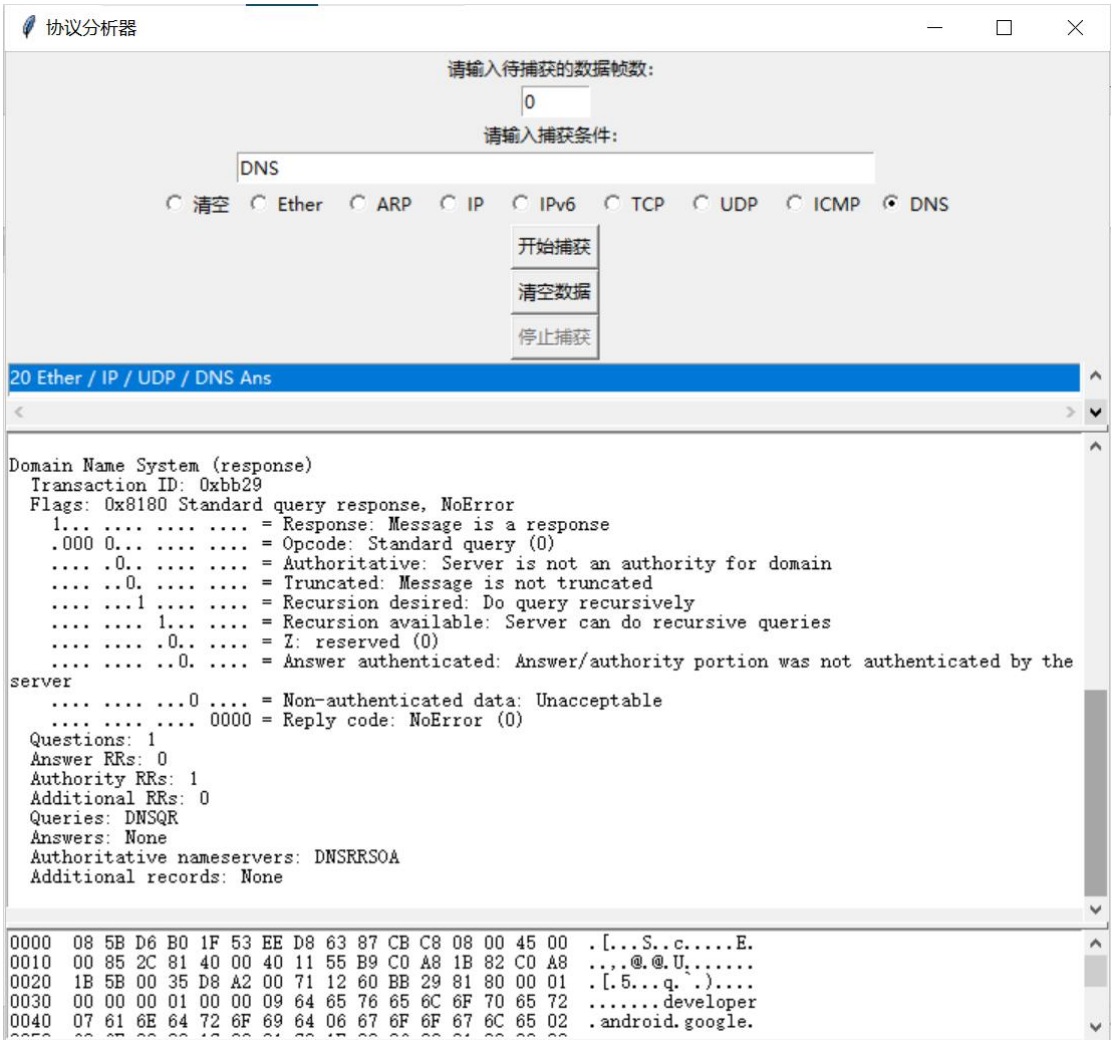

图 32 DNS 报文捕获与解析（下部分）

8 项目分析总结

8.1 遇到的问题及解决办法

8.1.1 项目重构中 tkinter 组件的多函数绑定和多参数传递问题

问题描述：在重构项目时，将界面与逻辑分开，因此 tkinter 组件的调用函数需要传参，产生了多函数绑定和多参数传递的问题，不正确的函数绑定和传参将导致编译不通过。

解决办法：经多方搜寻，发现可以使用 lambda 表达式进行多个函数的绑定。

8.1.2 关于 Stream index

问题描述：在 wires hark 中，经过抓包可以获得一个名为 Stream index 的值，不明白该值所代表的意义和获取方法。

解决方法：查找资料得到：the stream index is an internal Wireshark mapping to: [IP address A, TCP port A, IP address B, TCP port B]. All the packets for the same tcp.stream value should have the same values for these fields (though the src/dest will be switched for A->B and B->A packets). 即 Stream Index 是 wires hark 通过源地址、源端口和目的地址、目的端口映射出的一个值，相同的数据报具有相同的 Stream Index。由于未查找到它的映射规则，因此项目中省略了 Stream Index。

8.1.3 ICMP 报文中 Identifier 和 Sequence Number 的 BE、LE 之分

问题描述：对 ICMP 报文进行解析时，在 wires hark 中，Identifier 和 Sequence Number 分别由相同的二进制解析出 BE、BL 两种值，不明了其原因。

解决方法：wireshark 考虑到 window 系统与 Linux 系统发出的 ping 报文（主要指 ping 应用字段而非包含 IP 头的 ping 包）的字节顺序不一样（windows 为 LE：little-endian byte order，Linux 为 BE：big-endian），分别告诉信息，其本质内容是不变的，只是表达形式不同。由 Scapy 解析出的是大端字节序 BE，通过大端字节序转小端字节序的函数可以得到 LE。

8.1.4 解析 TCP 报文偶尔只显示 MAC 解析的内容，同时报出 Error

问题描述：捕获 TCP 报文后双击报文，只显示 MAC 解析部分的内容，同时在 TCP 解析函数处报出错误。

解决方法：这是由于只完成了 IP 报文的解析并在 TCP 报文中调用，当 TCP 报文中的是 IPv6 协议，就会因为无法解析 IPv6 出现错误。新增 IPv6 报文的解析函数，并在 TCP 报文的解析函数中判断报文里是 IP 还是 IPv6，并调用相应的函数，便解决了问题。

8.1.5 对 IPv6 报文捕获时，捕获的却全是 IP 报文

问题描述：在协议分析器中，对 IPv6 报文进行筛选捕获，只捕获到 IP 报文。

解决方法：仔细审查代码发现在回调函数 ip_monitor_callback(app, pkt)和报文双击调用函数 chosen_pdu_analysis(self, event)中，是以捕获条件输入框的字符串是否存在相应协议名称为判断条件的，而对 IP 的判断放在 IPv6 前面，因此满足存在字符串“IPv6”时必定满足存在“IP”，因而直接进行了 IP 的捕获和解析。将对 IPv6 的判断放到 IP 前面便可以解决问题。

8.1.6 将各报文中不同的值代表的含义通过字典保存和索引

问题描述：直接在报文分析函数中，通过 if 判断报文各个值不同的含义过于繁琐，使函数代码冗长。

解决方法：将各报文中不同的值代表的含义通过字典保存在公共常量模块，方便直接索引获取。

8.2 项目亮点及不足之处

8.2.1 亮点

(1) 代码结构清晰：将不同功能的函数分别放在不同的模块中，可以使代码结构更加清晰易读，方便理解、维护和使用。
(2) 模块化设计：将功能相似的函数放在同一个模块中，可以方便地进行模块化设计，提高代码的可重用性和可维护性。
(3) 易于扩展：将函数按照功能进行分类，可以方便地扩展新的功能，只需要在相应的模块中添加新的函数即可。
(4) 对界面添加便捷的单选组件，方便对于捕获条件的快速输入

8.2.2 不足

(1) 函数调用复杂度增加：将函数分散到不同的模块中，增加了函数调用的复杂度，需要在调用函数先进行模块导入。
(2) 模块间依赖关系增加：将不同功能的函数分别放在不同的模块中，会增加模块间的依赖关系，需要在模块间进行导入。

(3) 不利于全局变量的管理：将函数分散到不同的模块中，不利于全局变量的管理，需要在模块间进行变量的传递和更新。

9 课程设计总结

9.1 想法

通过完善一个基于 Scapy 的协议分析器，我学习到了很多：

(1) 网络协议的基础知识：深入理解 MAC、IP、ARP、TCP、UDP、ICMP、DNS 等网络协议的原理和工作方式，了解协议的各个字段和作用。

(2) Python 编程能力：通过使用 Python 和 Scapy 库，学习到 Python 编程在处理网络数据中的应用，提升 Python 编程能力。

(3) 网络嗅探和协议分析技术：了解并掌握网络嗅探和协议分析技术的原理和方法，通过实际操作培养解决网络问题的能力。

(4) 图形化界面设计：通过使用 Tkinter 库，学习到图形化界面设计的基础知识，掌握界面设计和实现的方法。

(5) 问题排查能力：通过该实践，我不断在发现问题和解决问题中提高自己的问题排查能力，并增加自己的知识储量。

9.2 建议

(1) 注重实践与理论结合：在实践过程中，需要注重理论与实践的结合，让学生理解协议分析器的工作原理和用途，以及在什么情况下使用。

(2) 逐步增加实践难度：在实践过程中，应该设置不同难度的任务，让学生由浅入深地学习和掌握相关知识。

(3) 鼓励创新和探索：鼓励学生进行创新和探索，发现并尝试解决新的问题，培养学生的创新能力和解决问题的能力。

参考文献

[1] 谢希仁.计算机网络[M].8.电子工业出版社,2021.

[2] Philippe Biondi <phil at secdev.org>.Scapy 2023.09.03 文档[EB/OL].[2023.09.16].https://www.osgeo.cn/scapy/introduction.html.

[3] <flutterdev@88.com>.Tkinter从入门到实战详解[EB/OL].[2023.09.16].http://w

ww.bczl.xyz/tkinter/doc/1%20%E6%A6%82%E8%BF%B0.html.
[4] <xujinzhong027@outlook.com>.ip、tcp、udp 首部校验和算法的 Python 实现[EB/OL].[2023.09.17].https://xujinzh.github.io/2022/01/15/ip-tcp-udp-header-checksum/index.html.
[5] pcent.Follow tcp stream - Where does field "Stream index" come from?[EB/OL].[2023.09.17].https://stackoverflow.com/questions/6076897/follow-tcp-stream-where-does-field-stream-index-come-from.
[6] JIngles123.icmp数据包BE、LE解释[EB/OL].[2023.09.17].https://blog.csdn.net/qq_40467670/article/details/83820606.
[7] huey2672.Wireshark - ICMP 报文分析[EB/OL].[2023.09.17].https://www.cnblogs.com/huey/p/4820998.html.
[8] damanchen.字节序、大端字节序(Big Endian)、小端字节序(Little Endian)总结[EB/OL].[2023.09.17].https://blog.csdn.net/damanchen/article/details/112424874.
[9] wikipedia.IPv6 packet[EB/OL].[2023.09.18].https://en.wikipedia.org/wiki/IPv6_packet.
[10] wikipedia.Differentiated services[EB/OL].[2023.09.18].https://en.wikipedia.org/wiki/Differentiated_services.
[11] wikipedia.Explicit Congestion Notification[EB/OL].[2023.09.18].https://en.wikipedia.org/wiki/Explicit_Congestion_Notification.
[12] Philippe Biondi <phil at secdev.org>.scapy.layers.dns[EB/OL].[2023.09.19].https://scapy.readthedocs.io/en/latest/api/scapy.layers.dns.html.
[13] wikipedia.Domain Name System[EB/OL].[2023.09.19].https://en.wikipedia.org/wiki/Domain_Name_System.
[14] <0@twfb.org>.通过Scapy学习网络知识[EB/OL].[2023.09.13].https://github.com/twfb/Scapy-Note#91ipscapylayersinetip.
[15] 鱼丸、粗面.Python tkinter 绑定多个函数 + 传递参数详解[EB/OL].[2023.09.14].https://blog.csdn.net/qq_34745941/article/details/117080144.
[16] frank909.Python多线程编程(一）：threading 模块 Thread 类的用法详解[EB/OL].[2023.09.19].https://blog.csdn.net/briblue/article/details/85101144.