华中科技大学计算机学院
《计算机通信与网络》实验报告

实验名称 使用网络协议分析仪 Wireshark

姓 名	班 级	学 号	得 分

教师评语：

一 实验环境

1. 实验环境：运行 Arch Linux x86_64 操作系统的 PC 机一台
2. 网络平台：校园网（HUST_WIRELESS）
3. IP 地址：10.14.118.185
4. Wireshark 版本：2.6.3

二 实验目的

1. 能够正确安装配置网络协议分析软件 Wireshark。
2. 熟悉使用 Wireshark 分析网络协议的基本方法，加深对协议格式、协议层次和协议交互过程的理解。

三 实验内容及步骤

3.1 软件安装

在 Arch Linux 下执行 pacman -Ss wireshark，可以看到有如下的安装包：

```
panyue@Saltedfish ~ pacman -Ss wireshark
community/wireshark-cli 2.6.3-1 [已安装]
    a free network protocol analyzer for Unix/Linux and Windows - CLI version
community/wireshark-common 2.6.3-1 [已安装]
    Common files used by wireshark-gtk and wireshark-qt
community/wireshark-gtk 2.6.3-1
    a free network protocol analyzer for Unix/Linux and Windows - GTK frontend
community/wireshark-qt 2.6.3-1 [已安装]
    a free network protocol analyzer for Unix/Linux and Windows - Qt frontend
panyue@Saltedfish ~
```

图 1 搜索 wireshark 的包

我们执行如下命令安装第一、二、四个即可。

sudo pacman -S wireshark-cli wireshark-common wireshark-qt

这里第三个和第四个分别是 wireshark 的 gtk 和 qt 编写的图形界面，选择一个即可。

3.2 使用 Wireshark 分析协议

打开 Wireshark，我们可以看到如下的界面。

界面中间显示的是现有的网卡，这里选择 wlp3s0（无线网卡），进行操作。

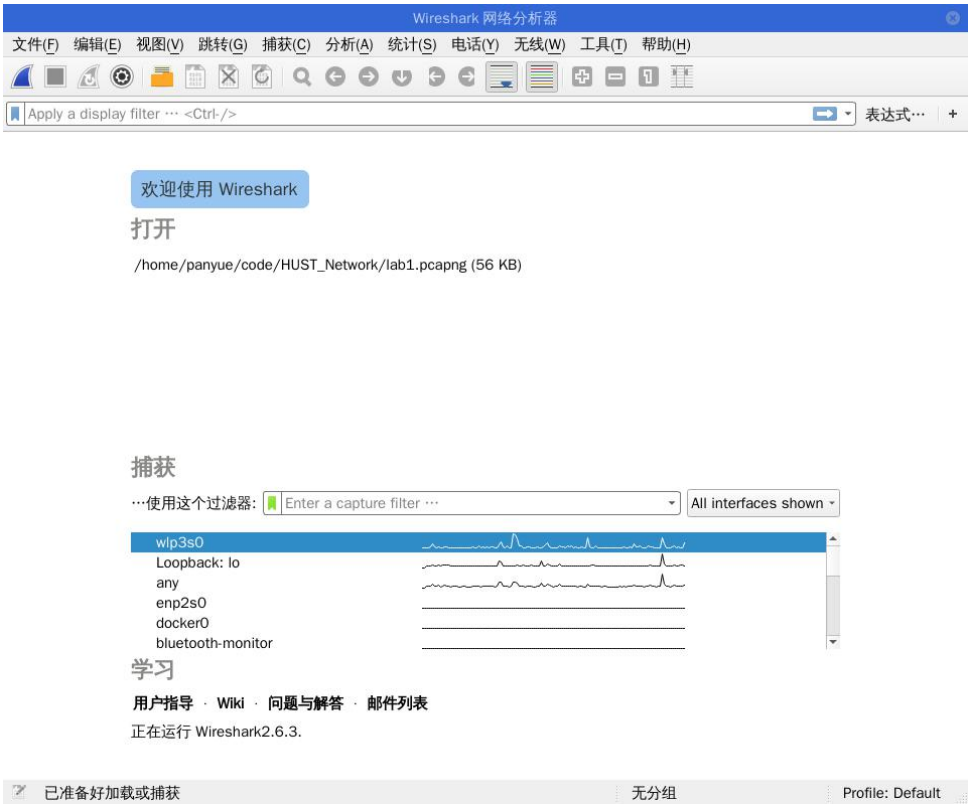


图 2 wireshark 主界面

实验的步骤如下：

1. 点击左上角“鲨鱼”形状按钮，开始俘获分组
2. 在命令行执行命令 traceroute www.baidu.com，等待程序执行完毕
3. traceroute 执行完毕后，结束俘获，将信息保存位 lab1.pcapng
4. 观察俘获到的任意分组，查看其各字段
5. 在筛选框中输入 icmp，筛选 icmp 包进行分析，并与 traceout 的输出进行比较。

四 实验结果

4.1 协议分析

随便点击一个分组，可以看到系统能够对俘获或者打开的踪迹文件中的分组信息进行分析，有编号、事件、源地址、目的地址、协议、长度和信息等列。最下面窗口中是对应所选分组以十六进制数和 ASCII 形式的内容。

这里选择 TCP 协议的源端口号，可以看到下面窗口中对应的十六进制位和 ASCII 形式的内容。并且从右侧内容中可以看出这是一个 HTTP POST 请求。

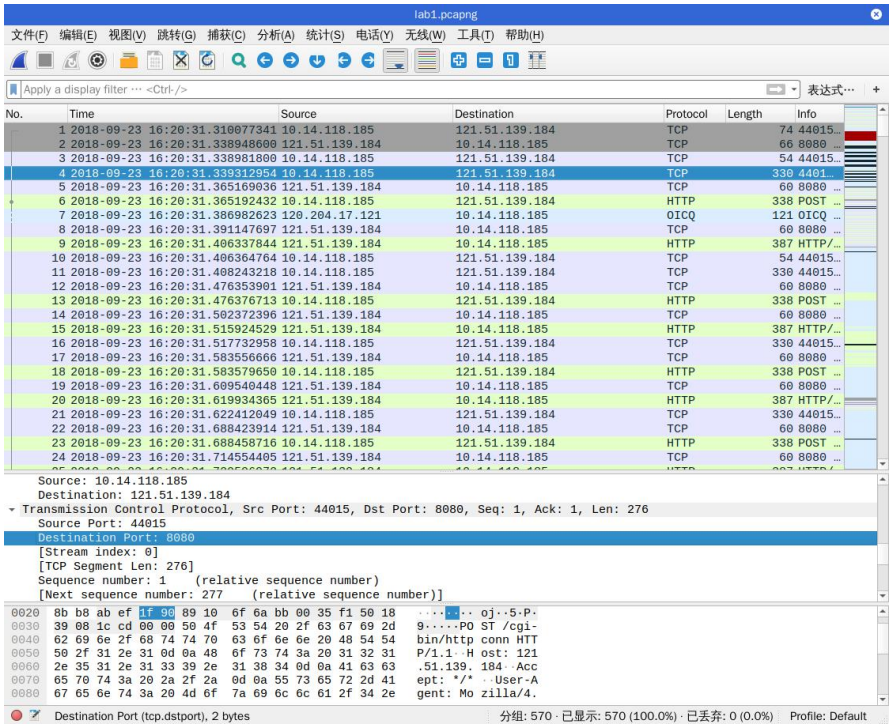


图 3 wireshark 协议分析

4.2 traceroute 结果分析

traceroute 的输出结果如下，显示为*的表示没有数据返回：

```
panyue@Saltedfish ~ traceroute www.baidu.com
traceroute to www.baidu.com (111.13.100.92), 30 hops max, 60 byte packets
 1  * * *
 2  192.168.243.33 (192.168.243.33)  4.958 ms  4.966 ms  4.961 ms
 3  192.168.243.129 (192.168.243.129)  4.955 ms  5.557 ms  5.558 ms
 4  111.47.18.1 (111.47.18.1)  25.458 ms  25.465 ms  25.459 ms
 5  * * *
 6  221.183.58.101 (221.183.58.101)  11.621 ms  6.706 ms  6.653 ms
 7  221.183.37.225 (221.183.37.225)  23.821 ms  23.816 ms  23.289 ms
 8  * * *
 9  111.13.98.93 (111.13.98.93)  23.886 ms 111.13.0.174 (111.13.0.174)  27.807 ms 111.13.98.93 (111.13.98.93)  24.618 ms
10  111.13.98.101 (111.13.98.101)  24.625 ms 111.13.98.93 (111.13.98.93)  24.459 ms 111.13.108.22 (111.13.108.22)  26.740 ms
11  111.13.112.61 (111.13.112.61)  28.846 ms 111.13.112.57 (111.13.112.57)  36.061 ms *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

图 4 traceroute 输出结果

通过网络上的 IP 位置查询工具，给出每一步的 IP 地理位置

1. * * * 三次均拒绝，推测为第一级路由器
2. 192.168.243.33 本地局域网
3. 192.168.243.129 本地局域网

4. 111.47.18.1 武汉
5. * * *
6. 221.183.58.101 中国移动
7. 221.183.37.225 中国移动
8. * * *
9. 111.13.98.93 111.13.0.174 111.13.98.93 北京市 中国移动
10. 111.13.98.101 111.13.98.93 111.13.108.22 北京市 中国移动
11. 111.13.112.61 111.13.112.57 * 北京市 中国移动
12. * * *

Wireshark 筛选出的 icmp 包如下图：

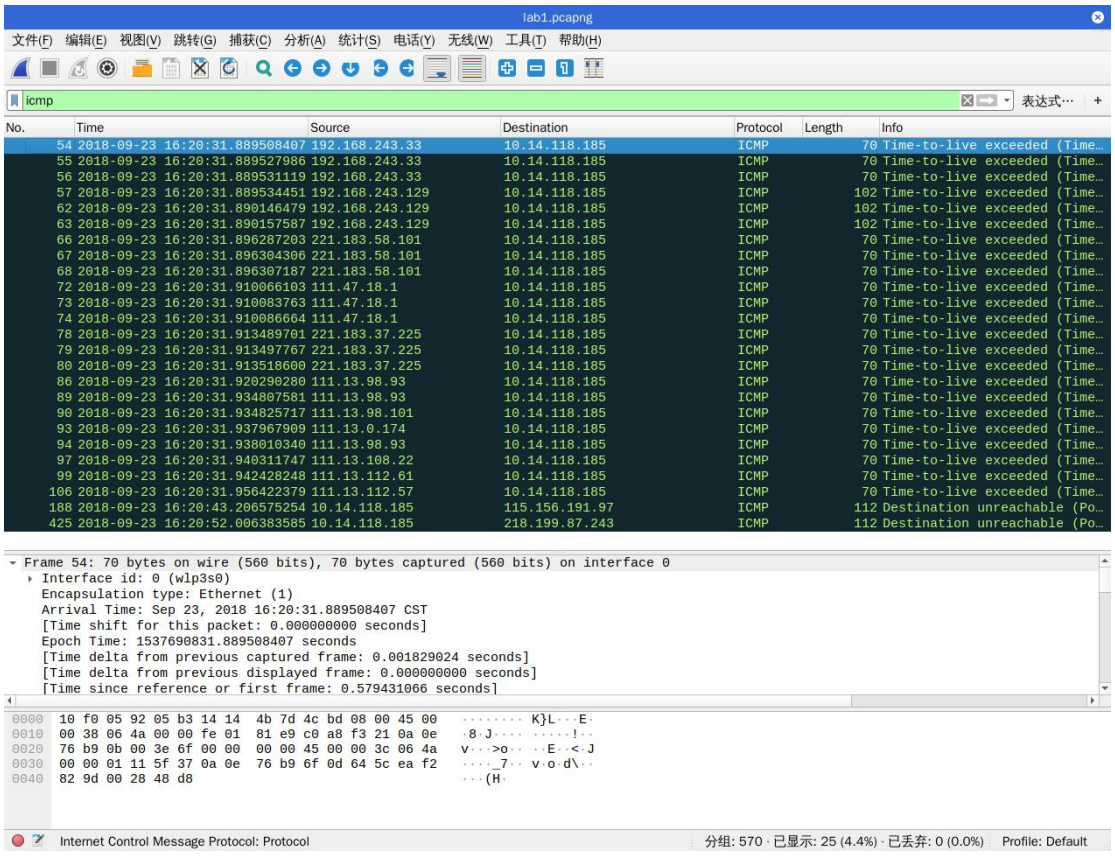


图 5 icmp 包俘获结果

仔细观察结果，可以看到俘获的包结果和 traceroute 的输出是一一对应的，traceroute 默认是发送三个分组，这里每一跳一般都有三个 icmp 包返回，出现*的第 11 级就只有两个返回，并且 IP 地址都是对应的，验证了实验的正确性。

五 实验中的问题及心得

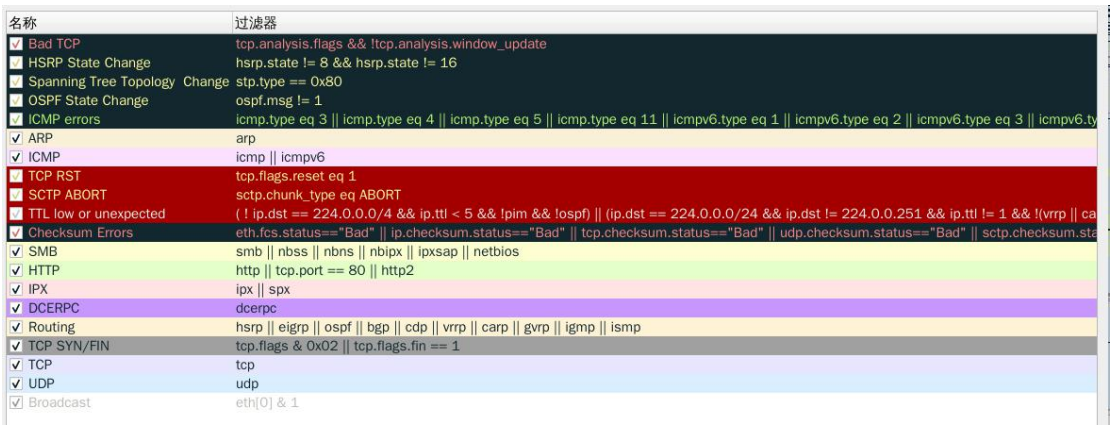
1.1 实验中的问题

1. 为什么有些级中出现了相同的 IP 地址？

traceroute 会发送三个分组，这些分组可能走了不同的路径，**111.13.98.93** 可能在第一个分组走的路径上是第九跳，在另一个分组走的路径上是第十跳，因此不同的级中出现了相同的 **IP** 地址。

2. Wireshark 中出现了很多不同的颜色，代表什么？

点击视图->着色规则，可以看到如下图所示的窗口：



名称	过滤器
Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
HSRP State Change	hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Change	stp.type == 0x80
OSPF State Change	ospf.msg != 1
ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 || icmpv6.ty
ARP	arp
ICMP	icmp || icmpv6
TCP RST	tcp.flags.reset eq 1
SCTP ABORT	sctp.chunk_type eq ABORT
TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || ca
Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.checksum.status=="Bad" || udp.checksum.status=="Bad" || sctp.checksum.sta
SMB	smb || nbss || nbns || nbipx || ipxsap || netbios
HTTP	http || tcp.port == 80 || http2
IPX	ipx || spx
DCERPC	dcerpc
Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp
TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
TCP	tcp
UDP	udp
Broadcast	eth[0] & 1

图 6 Wireshark 着色规则

从图中我们就可以看出，红色的往往是一些丢弃的包，黑色是错误的包，浅色的一般是正常的包等等，熟练记住了颜色分类有助于更高效地使用 **Wireshark**。

5.2 实验总结

本次实验练习 **Wireshark** 的使用，自己以前也玩过这方面的一些操作，并不难。总之加深了对网络协议的理解，熟悉了一次网络请求的整个过程，为之后的实验做好了准备。