重庆大学大数据与软件学院

上 机 实 验 报 告

上机实践项目 IP 协议分析

课程名称 计算机网络

姓名 XXX 成绩

学号 2021XXXX 教师 **XXX**

班级 软工 X 班 日期 2023/4/25

《计算机网络》上机实验报告

开课实验室：DS1502 **2023 年 4 月 25 日**

姓　名	XXX	年级、班级	2021 级软件工程 X 班	学号	**2021XXXX**
上机（项目）名称		IP 协议分析		指导教师	XXX
教师评语					

教师签名：

年　月　日

一、上机目的

In this lab, we'll investigate the IP protocol, focusing on the IP datagram.

•We'll do so by analyzing a trace of IP datagrams sent and received by an execution of the traceroute program (the traceroute program itself is explored in more detail in the Wireshark ICMP lab).

• We'll investigate the various fields in the IP datagram, and study IP fragmentation in detail. • we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action;

Before beginning this lab, you'll probably want to review sections 1.4.3 in the text1 and section 3.4 of RFC 2151 [ftp://ftp.rfc-editor.org/in-notes/rfc2151.txt] to update yourself on the operation of the traceprogram.

You'll also want to read Section 4.4 in the text, and probably also have RFC 791 [ftp://ftp.rfc-editor.org/in-notes/rfc791.txt] on hand as well, for a discussion of the IP protocol.

二、基本原理

Wireshark: Wireshark is a popular network protocol analysis tool that captures network traffic and analyzes the communication process of various protocols. It is a free and open-source software that can run on multiple operating systems including Windows, Mac OS X, and Linux. Wireshark captures packets from a network interface and can analyze them in real-time or offline. It supports various protocols such as TCP, UDP, IP, HTTP, FTP, DNS, ICMP, and can also decode TLS and SSL encrypted traffic. By using Wireshark, users can perform deep analysis of network communication processes, including understanding the data type, protocol header information, packet transmission time, and relationships between different packets. This is extremely useful for network administrators and security professionals.

IP: IP (Internet Protocol) is a fundamental protocol used to transmit data over the internet. It operates at the Network layer (Layer 3) of the OSI model and is responsible for routing packets from the source device to the destination device across one or multiple networks. In general, IP provides a standardized way for devices to communicate with each other over the internet by uniquely identifying each device on the network with an IP address. When you send data over the internet, it is broken down into smaller pieces called packets. Each packet is tagged with the source and destination IP addresses so that routers and other network devices can direct the packets to the appropriate destination. There are two main versions of the IP protocol in use today: IPv4 and IPv6. IPv4 uses 32-bit addresses and supports up to 4.3 billion unique addresses, which are now becoming scarce due to the rapid growth of the internet and the proliferation of connected devices. IPv6, on the other hand, uses 128-bit addresses and can support an astronomical number of unique addresses, ensuring that there will be enough addresses for the foreseeable future.

PingPlotter: PingPlotter is a network diagnostic tool that uses tracing and monitoring to identify and isolate issues with networks and internet connections. It tracks network performance by sending packets (pings) to a target device or server and then graphically displays the results over time. With PingPlotter, you can measure and visualize various aspects of network performance such as latency, packet loss, and jitter. The tool also provides real-time data about the route that packets take between your computer and the target device or server, allowing you to pinpoint exactly where in the network issues are occurring. PingPlotter has both a graphical user interface and a command-line interface, making it a flexible tool for both novice and advanced users. It also allows you to save and share data with others in a variety of formats, including CSV and PDF. In addition to its basic functionality, PingPlotter offers several advanced features, such as automated alerting when network conditions exceed predefined thresholds and the ability to run multiple traces simultaneously. Overall, PingPlotter is a reliable and comprehensive tool for diagnosing and troubleshooting network problems, and it is widely used by network administrators, IT professionals, and gamers alike.

Windows: Here are the steps to use Wireshark software on Windows11 for analyzing IP protocol:

- Download and install Wireshark on your Windows 11 computer.
- Launch Wireshark from the Start menu or desktop shortcut.

• Select the network interface you want to capture packets from by clicking the "Capture Options" button in the toolbar. You can choose from a list of available interfaces, such as Ethernet or Wi-Fi.

• Click the "Start" button to begin capturing packets on the selected interface.

• Use the display filter field at the top of the screen to filter the captured packets to only show IP traffic. Simply type "ip" into the filter field and press Enter.

• You can now view the details of each IP packet that has been captured by selecting it in the packet list. The details pane will show information about the packet such as the source and destination IP addresses, the protocol being used, and any payload data.

• You can also use Wireshark's analysis tools to investigate specific aspects of the IP traffic, such as the volume of traffic from a particular IP address or protocol.

• Once you have finished analyzing the IP traffic, stop the capture by clicking the "Stop" button in the toolbar.

Overall, Wireshark is a powerful tool for analyzing IP traffic in Windows 11 and can help you diagnose network issues and optimize network performance.

三、使用的软件、硬件

Software: Windows11 && Wireshark && PingPlotter && Firefox;

Hardware: Lenovo Legion R9000P2021H.

四、上机操作步骤

1. Capturing packets from an execution of traceroute

In order to generate a trace of IP datagrams for this lab, we'll use the traceroute program to send datagrams of different sizes towards some destination, X. Recall that traceroute operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by at least one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing traceroute) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing traceroute can learn the identities of the routers between itself and destination X by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

We'll want to run traceroute and have it send datagrams of various lengths.

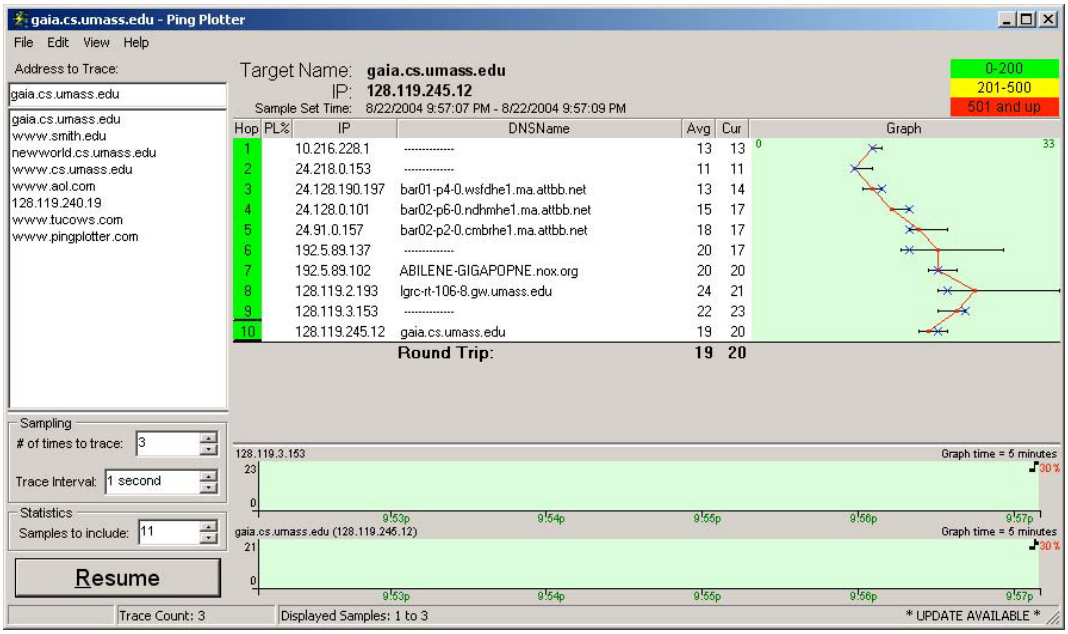
- **Windows.** The `tracert` program (used for our ICMP Wireshark lab) provided with Windows does not allow one to change the size of the ICMP echo request (ping) message sent by the `tracert` program. A nicer Windows `traceroute` program is *pingplotter*, available both in free version and shareware versions at http://www.pingplotter.com. Download and install *pingplotter*, and test it out by performing a few traceroutes to your favorite sites. The size of the ICMP echo request message can be explicitly set in *pingplotter* by selecting the menu item *Edit->Options->Packet Options* and then filling in the *Packet Size* field. The default packet size is 56 bytes. Once *pingplotter* has sent a series of packets with the increasing TTL values, it restarts the sending process again with a TTL of 1, after waiting *Trace Interval* amount of time. The value of *Trace Interval* and the number of intervals can be explicitly set in *pingplotter*.
- **Linux/Unix/MacOS.** With the Unix/MacOS `traceroute` command, the size of the UDP datagram sent towards the destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the `traceroute` command line immediately after the name or address of the destination. For example, to send `traceroute`datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

```
%traceroute gaia.cs.umass.edu 2000
```

Do the following:

1) Start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen (we'll not need to select any options here).

2) If you are using a Windows platform, start up pingplotter and enter the name of a target destination in the "Address to Trace Window." Enter 3 in the "# of times to Trace" field, so you don't gather too much data. Select the menu item Edit >Advanced Options->Packet Options and enter a value of 56 in the Packet Size field and then press OK. Then press the Trace button. You should see a pingplotter window that looks something like this:



Next, send a set of datagrams with a longer length, by selecting Edit->Advanced Options->Packet Options and enter a value of 2000 in the Packet Size field and then press OK. Then press the Resume button.

Finally, send a set of datagrams with a longer length, by selecting Edit¬>Advanced Options->Packet

Options and enter a value of 3500 in the Packet Size field and then press OK. Then press the Resume button.

Stop Wireshark tracing.

3) If you are using a Unix or Mac platform, enter three traceroute commands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

Stop Wireshark tracing.

If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's Windows computers[2]. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

2. A look at the captured trace

- In your trace, you should be able to see the series of ICMP Echo Request (in the case of Windows machine) or the UDP segment (in the case of Unix) sent by your computer and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers.
- In the questions below, we'll assume you are using a Windows machine; the corresponding questions for the case of a Unix machine should be clear.
- Whenever possible, when answering a question below you should hand in a printout of the packet(s) within the trace that you used to answer the question asked.
- When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).
- To print a packet, use File->Print, choose Selected packet only, choose Packet summary line, and select the minimum amount of packet detail that you need to answer the question.

1) Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

2) Within the IP packet header, what is the value in the upper layer protocol field?

3) How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

4) Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

- Next, sort the traced packets according to IP source address by clicking on the Source column header; a small downward pointing arrow should appear next to the word Source. If the arrow points up, click on the Source column header again.
- Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol portion in the "details of selected packet header" window.
- In the "listing of captured packets" window, you should see all of the subsequent ICMP messages (perhaps with additional interspersed packets sent by other protocols running on your computer) below this first ICMP.
- Use the down arrow to move through the ICMP messages sent by your computer.

5) Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?

6) Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?

7) Describe the pattern you see in the values in the Identification field of the IP datagram.

- Next (with the packets still sorted by source address) find the series of ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router.

8) What is the value in the Identification field and the TTL field?

9) Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

- **Fragmentation**
- Sort the packet listing according to time again by clicking on the Time column.

10) Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram? [Note: if you find your packet has not been fragmented, you should download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the ip-ethereal-trace-1packet trace. If your computer has an Ethernet interface, a packet size of 2000 should cause fragmentation.3]

11) Print out the first fragment of the fragmented IP datagram. What information in the IP header

indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

12) Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?

13) What fields change in the IP header between the first and second fragment?

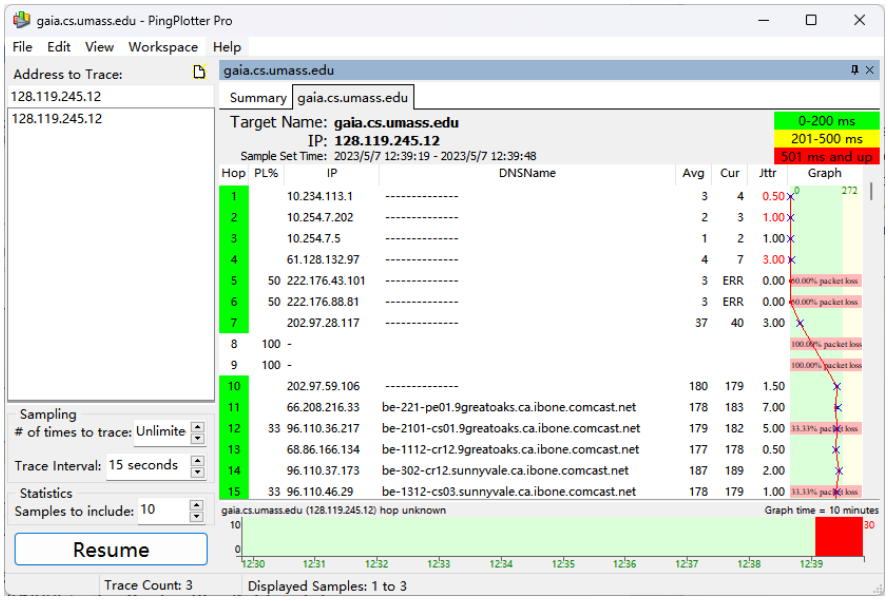
- Now find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 3500.

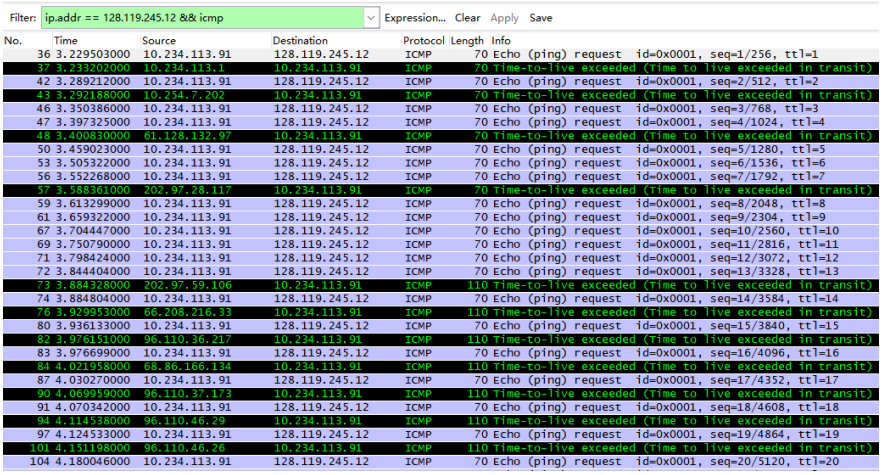
14) How many fragments were created from the original datagram?

15) What fields change in the IP header among the fragments?

五、过程原始记录(数据、图表、计算等)

1. Capturing packets from an execution of traceroute





2. A look at the captured trace

Q1: Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

A1: 我的 IP 地址为：10.234.113.91；

```
⊞ Frame 36: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 1
    Protocol: ICMP (1)
  ⊟ Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
      [Good: False]
    ⊞ [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x3b62 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ⊞ Data (28 bytes)
```

```
0000  bc 3f 8f dc f9 c9 38 f3  ab cd 80 31 08 00 45 00   .?....8. ...1..E.
0010  00 38 a4 ff 00 00 01 01  00 00 0a ea 71 5b 80 77   .8...... ....q[.w
0020  f5 0c 08 00 3b 62 00 01  00 01 30 45 50 69 6e 67   ....;b.. ..0EPing
0030  50 6c 6f 74 74 65 72 50  72 6f 33 2e 34 30 2e 32   PlotterP ro3.40.2
0040  70 30 45 50 69 6e                                 p0EPin
```

Q2: Within the IP packet header, what is the value in the upper layer protocol field?

A2: 上层协议字段的值为 1，表明为 ICMP；

```
⊞ Frame 36: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 1
    Protocol: ICMP (1)
  ⊟ Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
      [Good: False]
    ⊞ [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

Q3: How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

A3: IP 首部长度为 20bytes,又因为总长度为 56bytes,所以有效载荷为 36bytes（56-20=36 bytes）；

```
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 1
    Protocol: ICMP (1)
  ⊟ Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
      [Good: False]
    ⊞ [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

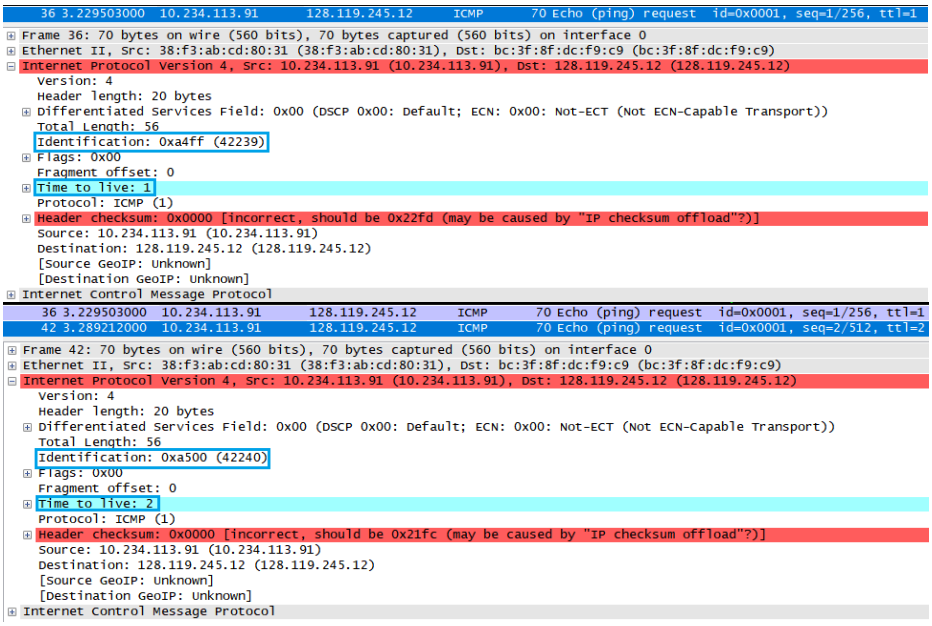
Q4: Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

A4: 这个 IP 数据报没有被分片，因为 Flags=0，Fragment offset=0；

```
⊞ Frame 36: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 1
    Protocol: ICMP (1)
  ⊟ Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
      [Good: False]
    ⊞ [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

Q5: Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?

A5: 在我的计算机发送的这一系列 ICMP 消息中，IP 数据报中总是更改的字段为 Identification 和 Time to live（TTL）；



Q6: Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?
A6: 在一系列 ICMP 信息中：

以下字段保持不变：

- 版本号（Version）
- 头部长度（Header Length）
- 区分服务（Type of Service）
- 总长度（Total Length）
- 标志位（Flags）
- 片偏移（Fragment Offset）
- 协议（Protocol）

以下字段必须保持不变：

- 源 IP 地址（Source IP Address）
- 目的 IP 地址（Destination IP Address）
- 校验和（Checksum）：ICMP 报文内容发生改变时，IP 头部检验和应该重新计算，但因为它只涉及到 IP 头部，所以此字段不会被更改；

以下字段必须更改：

- 标识（Identification）：用来唯一地标识一个报文的所有分片；
- 生存次数（Time to Live）：每个 ICMP Echo Request 消息都包含一个 TTL 值，用于限制数据包在网络中传输的最大跳数。该值将随着每个报文的发送而减少，以反映它在网络中的路径。

需要注意的是，除了以上列出的字段外，ICMP Echo Request 和后续的 ICMP 消息还包含其他一些特定于 ICMP 的字段，如类型（Type）、代码（Code）、校验和（Checksum）和序列号（Sequence Number），这些字段也会根据具体情况而改变。

```
    36 3.229503000  10.234.113.91      128.119.245.12      ICMP      70 Echo (ping) request  id=0x0001, seq=1/256, ttl=1
Frame 36: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  Flags: 0x00
    Fragment offset: 0
  Time to live: 1
    Protocol: ICMP (1)
  Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
      [Good: False]
    [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x3b62 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  Data (28 bytes)
```

```
    42 3.289212000  10.234.113.91      128.119.245.12      ICMP      70 Echo (ping) request  id=0x0001, seq=2/512, ttl=2
Frame 42: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa500 (42240)
  Flags: 0x00
    Fragment offset: 0
  Time to live: 2
    Protocol: ICMP (1)
  Header checksum: 0x0000 [incorrect, should be 0x21fc (may be caused by "IP checksum offload"?)]
      [Good: False]
    [Bad: True]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x3b61 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 2 (0x0002)
    Sequence number (LE): 512 (0x0200)
  Data (28 bytes)
```

Q7: Describe the pattern you see in the values in the Identification field of the IP datagram

A7: IP 数据报的标识字段是 16 位长的字段，用于标识一个数据报。在观察不同的 IP 数据报中的标识字段值时，存在一些模式；

首先，标识字段通常是按顺序递增的。也就是说，每个新的数据报的标识字段值通常比前一个更高。这是因为每个数据报都应该具有唯一的标识符，以便在传输过程中进行区分。

其次，当发送方将一个较大的数据报分割成多个小数据报进行传输时，这些小数据报的标识字段通常是一样的，只有片偏移字段和 MF（More Fragments）标志不同。这是因为它们都属于同一个原始数据报，并且需要使用相同的标识符来进行重组。

最后，在某些情况下，发送方可能会使用随机的标识符值来隐藏自己的身份，防止被攻击者识别出来。

```
    36 3.229503000  10.234.113.91      128.119.245.12      ICMP      70 Echo (ping) request  id=0x0001, seq=1/256, ttl=1
    42 3.289212000  10.234.113.91      128.119.245.12      ICMP      70 Echo (ping) request  id=0x0001, seq=2/512, ttl=2

⊞ Frame 36: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa4ff (42239)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 1
    Protocol: ICMP (1)
  ⊞ Header checksum: 0x0000 [incorrect, should be 0x22fd (may be caused by "IP checksum offload"?)]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
    42 3.289212000  10.234.113.91      128.119.245.12      ICMP      70 Echo (ping) request  id=0x0001, seq=2/512, ttl=2

⊞ Frame 42: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31), Dst: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9)
⊟ Internet Protocol Version 4, Src: 10.234.113.91 (10.234.113.91), Dst: 128.119.245.12 (128.119.245.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0xa500 (42240)
  ⊞ Flags: 0x00
    Fragment offset: 0
  ⊞ Time to live: 2
    Protocol: ICMP (1)
  ⊞ Header checksum: 0x0000 [incorrect, should be 0x21fc (may be caused by "IP checksum offload"?)]
    Source: 10.234.113.91 (10.234.113.91)
    Destination: 128.119.245.12 (128.119.245.12)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

Q8: What is the value in the Identification field and the TTL field?

A8: Identification 为 0x099b (2459)， TTL 为 255；

```
No.     Time           Source          ▲ Destination       Protocol Length Info
    37 3.233202000  10.234.113.1        10.234.113.91       ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)

⊞ Frame 37: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: bc:3f:8f:dc:f9:c9 (bc:3f:8f:dc:f9:c9), Dst: 38:f3:ab:cd:80:31 (38:f3:ab:cd:80:31)
⊟ Internet Protocol Version 4, Src: 10.234.113.1 (10.234.113.1), Dst: 10.234.113.91 (10.234.113.91)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 56
    Identification: 0x099b (2459)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: ICMP (1)
  ⊞ Header checksum: 0xb939 [correct]
    Source: 10.234.113.1 (10.234.113.1)
    Destination: 10.234.113.91 (10.234.113.91)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

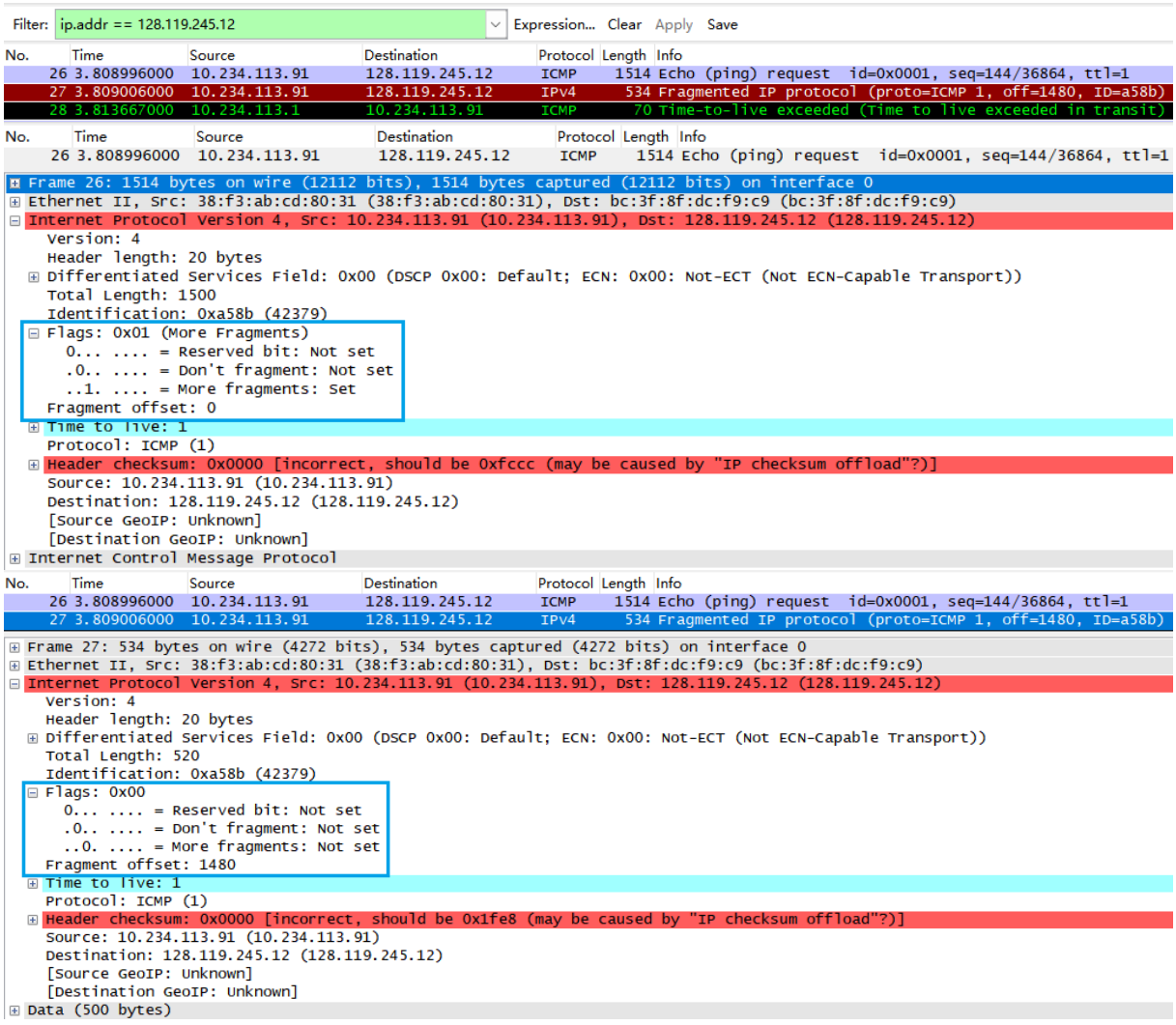
Q9: Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

A9: 在最近的路由器向我的计算机发送的所有超出 ttl 的 ICMP 应答中，Identification 均发生变化，TTL 均不变；

因为 Identification 字段通常是按顺序递增的，也就是说，每个新的数据报的标识字段值通常比前一个更高，每个数据报都应该具有唯一的标识符，以便在传输过程中进行区分，所以会变化，而这些信息均是从同一路由器经相同路径向我的计算机发送的，除非网络发生重大变化，则 TTL 不会发生改变；

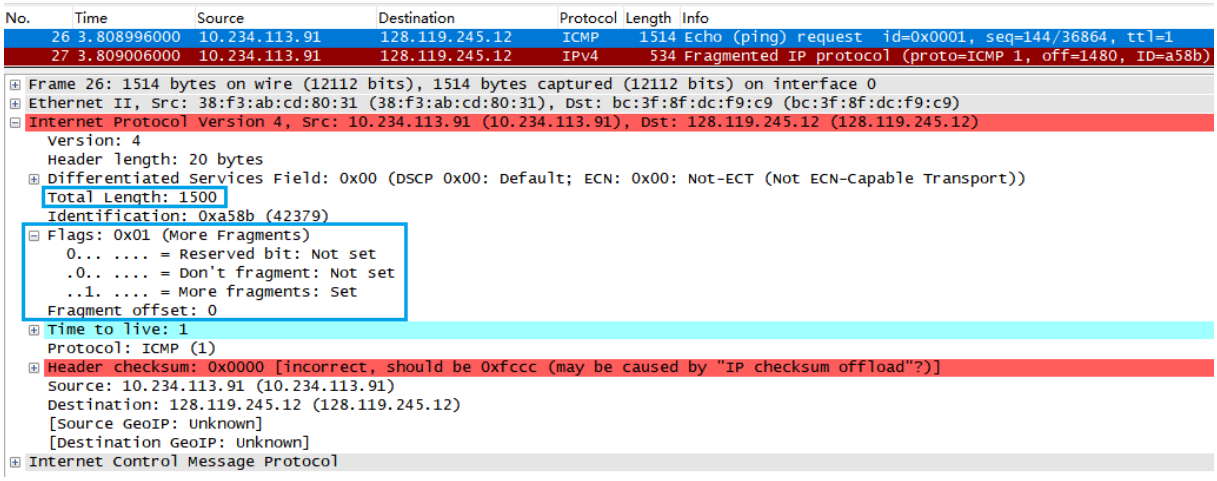
Q10: Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram? [Note: if you find your packet has not been fragmented, you should download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the ip¬ethereal-trace-1packet trace. If your computer has an Ethernet interface, a packet size of 2000 should cause fragmentation.3]

A10: 该消息被分散到多个 IP 数据报中；



Q11: Print out the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

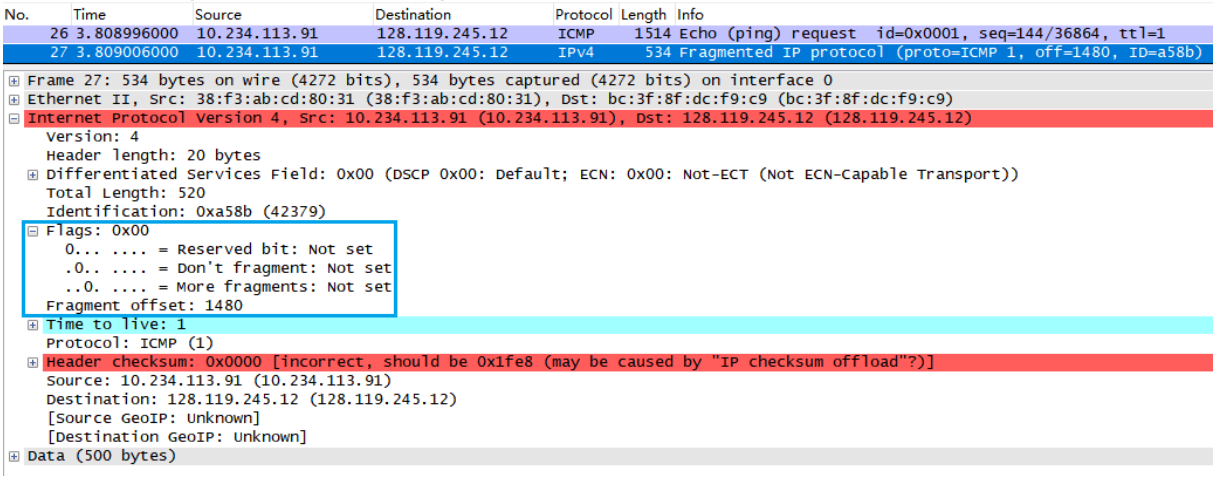
A11: Flags 为 0x01 表明该数据报已经被分片了， Fragment offset 为 0 说明这是第一个片段，这个数据报长度为 1500；



Q12: Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?

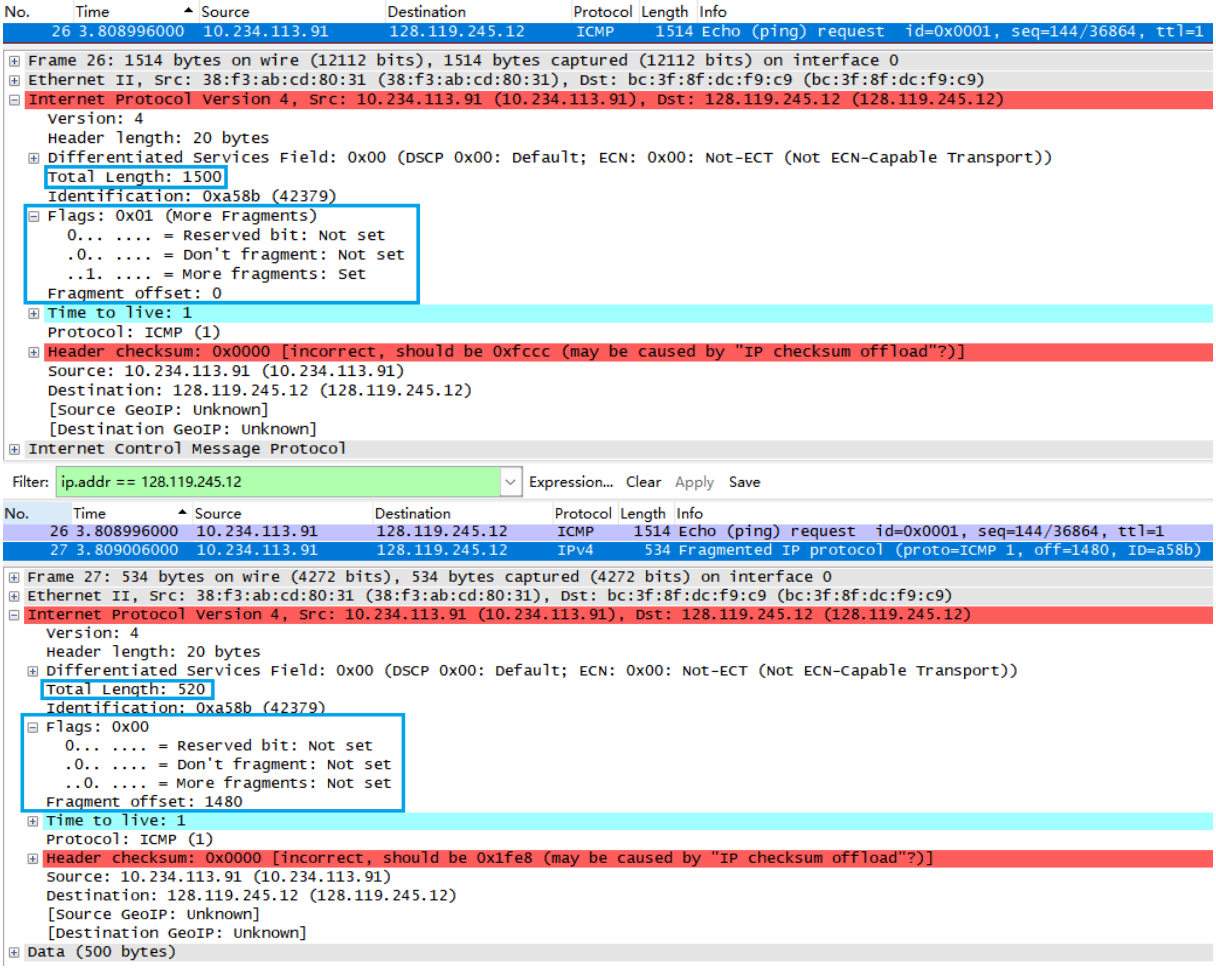
A12: Fragment offset 不为 0 而是 1480，表明这不是第一个分片；

又因为 Flags 为 0x00，More fragment=0，表明这是最后一个分片，后面没有更多分片了；



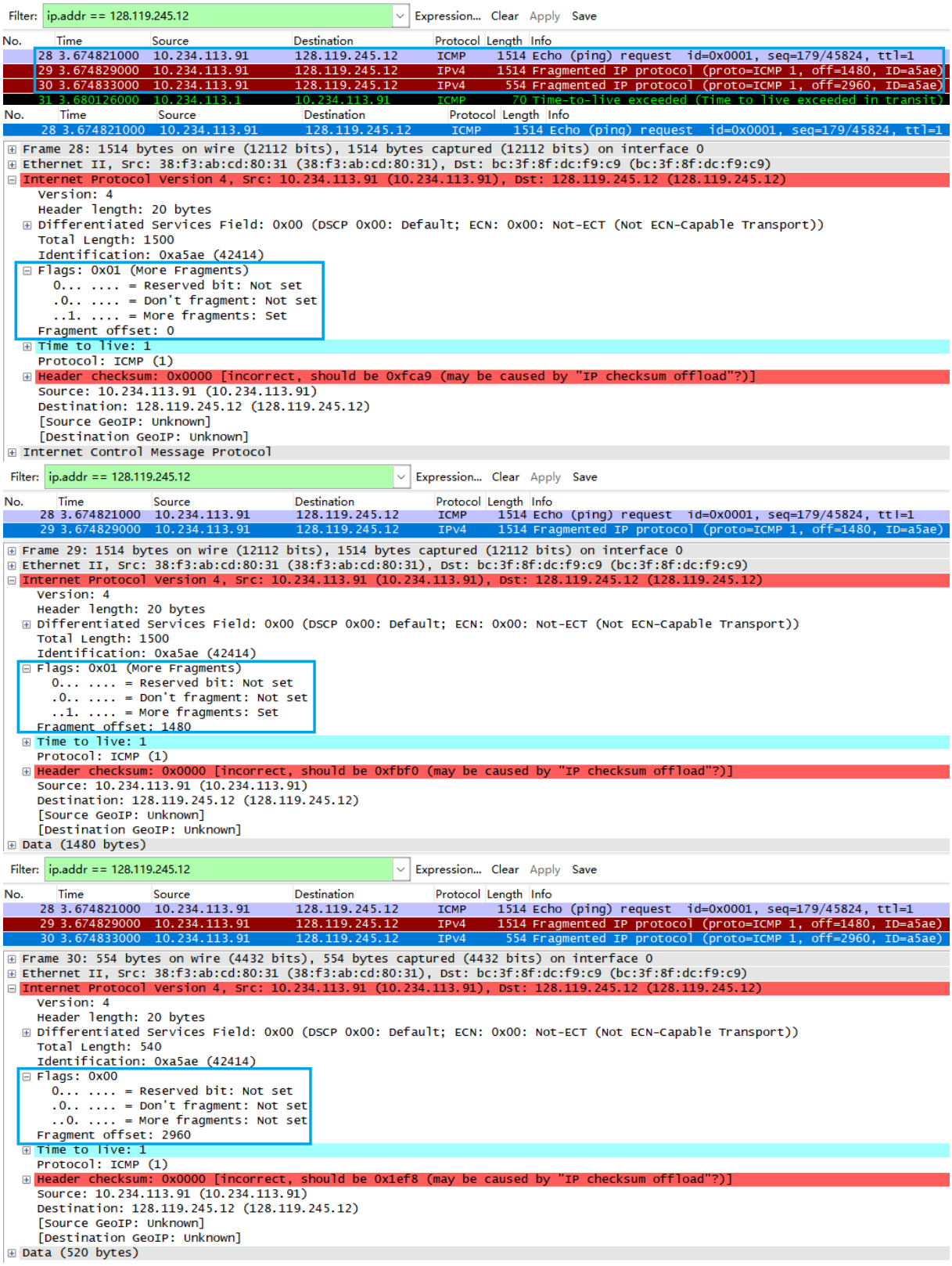
Q13: What fields change in the IP header between the first and second fragment?

A13: 在第一个和第二个片段之间的 IP 首部中 Total Length、Flags、Fragment offset 发生变化；



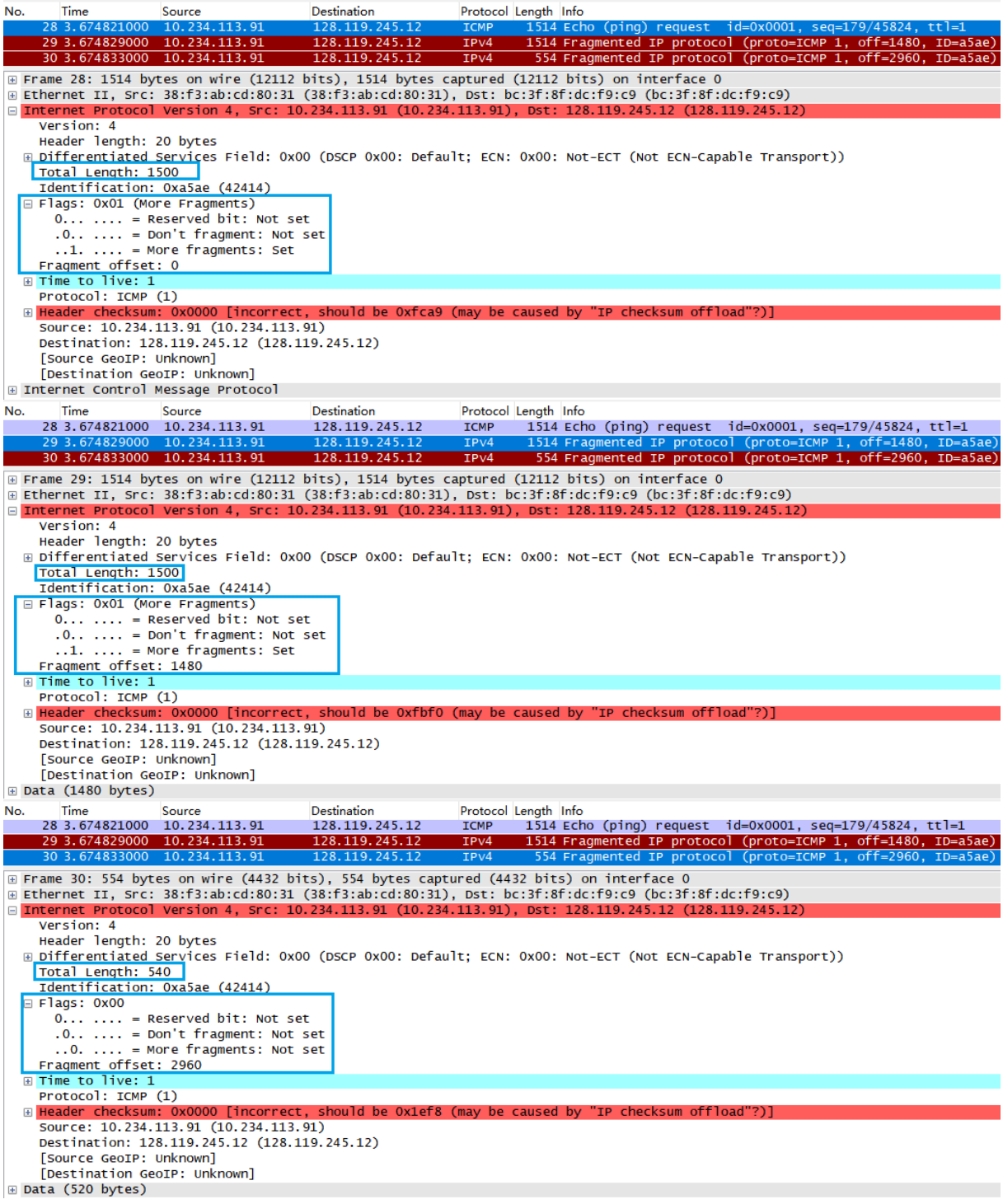
Q14: How many fragments were created from the original datagram?

A14: 从原始数据报创建了 3 个片段；



Q15: What fields change in the IP header among the fragments?

A15: IP 首部中 Total Length、Flags、Fragment offset 发生变化；



六、结果及分析

1. Capturing packets from an execution of traceroute

该部分旨在通过使用 traceroute 程序向某个目的地发送具有不同大小的数据报，生成 IP 数据报跟踪记录。traceroute 程序首先将 TTL 字段设置为 1，然后发送一个或多个数据报；然后将 TTL 值设置为 2，向相同的目标发送一系列一个或多个数据报；然后将 TTL 值设置为 3，向相同的目标发送一系列数据报，如此等等。每次路由器收到一个数据报时，都会将 TTL 减 1 并检查它是否为 0。如果 TTL 为 0，则路由器将向发送方返回一个 ICMP 消息（类型 11-TTL 超时）。因此，具有 TTL 为 1 的数据报会导致距离发送方一个路由器发送一个 ICMP TTL 超时消息；具有 TTL 为 2 的数据报将导致两个路由器发送 ICMP 消息回到发送方；具有 TTL 为 3 的数据报将导致三个路由器发送 ICMP 消息回到发送方，以此类推。通过这种方式，执行 traceroute 的主机可以通过查看包含 ICMP TTL 超时消息的数据报中的源 IP 地址，了解自己和目标 X 之间的路由器。

通过本实验，我可以深入了解和理解 TTL 字段在 IP 协议中的作用和意义，并了解网络跟踪技术的原理和方法。同时，还可以掌握 Wireshark 工具的使用，帮助我们更好地进行网络故障排除和监控。

2. A look at the captured trace

我们可以使用 traceroute 程序向目标发送具有不同大小的数据报来生成 IP 数据报跟踪记录，从而深入了解和理解 TTL 在 IP 协议中的作用和意义，以及网络跟踪技术的原理和方法。

通过 Wireshark 工具的使用，我们可以更好地进行网络故障排除和监控。

在分析 IP 数据包时，我们需要关注 IP 地址、协议字段、IP 头部长度、负载长度、标识字段等各个字段的含义和作用，以判断数据包是否被分片、哪些字段会改变等。

我们可以通过对 ICMP TTL 超时消息和 ICMP 回显请求消息的跟踪记录，了解自己和目标之间的路由器数量和路径。

七、上机实验总结

通过本次实验，我学习了如何通过抓包分析，了解和掌握 IP 协议中的各个字段及其作用。

在跟踪记录中，我们可以看到由计算机发送的 ICMP Echo 请求（Windows 机器）或 UDP 段（Unix 机器），以及由中间路由器返回给计算机的 ICMP TTL 超时消息。回答问题时，需要查看相关数据包，比如 IP 地址、协议字段、IP 头长度、有效负载大小等信息，并能够判断数据报是否被分片。

在问题中，我们需要利用 Wireshark 工具来进行抓包分析，从而回答各种与 IP 协议有关的问题。例如，为了找到由最近路由器发送给计算机的 ICMP TTL 超时回复，我们需要按源地址将跟踪记录的数据包进行排序。此外，为了确定一个 IP 数据报是否被分片，我们需要查看 Internet Protocol 部分中的 DF 标志位和 MF 标志位。

在最后一部分中，我们需要找到第一次更改 pingplotter packet size 后计算机发送的第一个 ICMP Echo 请求消息，并查看这个数据包是否被分片，以及其中哪些字段发生了变化。

总之，通过对 Wireshark 工具的使用，我们可以深入了解和理解 IP 协议的细节和原理，有助于我们更好地理解网络通信过程并进行网络故障排除。