

# Undergraduate Lab Report of Jinan University

Course Title Computer Networks Lab Evaluation                     

Lab Name IP Fragmentation

Lab Address N117 Instructor Dr. CUI Lin (崔林)

Student Name 蒋云翔 Student No 2022102330

College International School

Department                      Major CST

Date 2024 / 11 / 10

## 1. Introduction

### 1) Objective

- Understand the concept of MTU
- Understand the process of IP Fragmentation

### 2) Experiment Principle

#### 1. The format of IP packet

The IP packet header format offers a structured layout crucial for accurate data routing and processing over the Internet. This header includes fields specifying the protocol version, header length, quality of service parameters, packet lifespan, error-checking features, and the source and destination addresses. Additionally, there are options for specific settings in certain cases.

The figure below illustrates all essential information for an IP header format.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP				ECN				Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

Figure 1: The format of IP packet

## 2. IP Fragmentation

A datagram can traverse multiple networks, with each router along the path carefully decapsulating the IPv4 datagram from the incoming frame, processing it, and then re-encapsulating it into a new frame. The format and size of the outgoing frame depend on the protocols used by the underlying physical network. For example, when a router links a LAN to a WAN, it receives a frame in the LAN's format and transmits it in a format compatible with the WAN. This detailed process highlights the router's essential role in enabling seamless data transmission across different network types, efficiently managing format conversions and encapsulation requirements.

### 2.1 Maximum Transfer Unit (MTU)

Data link layer protocols each have a unique frame format, a characteristic feature across most protocols. Within these formats is a specific field that defines the maximum data field size. Essentially, when encapsulating a datagram in a frame, the total datagram size must stay within this specified limit. This limitation is determined by the constraints of both the hardware and software operating within the network, ensuring smooth and efficient data transmission.

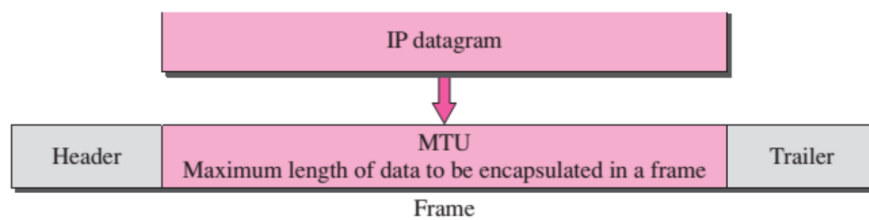


Figure 2: MTU

The value of the MTU depends on the physical network protocol. Following table shows the values for some protocols.

Protocol	MTU	Protocol	MTU
Hyperchannel	65,535	Ethernet	1,500
Token Ring(16Mbps)	17,914	X.25	576
Token Ring(4Mbps)	4,464	PPP	296
FDDI	4,352	–	–

Figure 3: MTUs in different networks

To ensure that the IPv4 protocol remains independent of physical network limitations, its designers set a maximum allowable datagram length of 65,535 bytes.

This limit enhances transmission efficiency, especially when paired with protocols that support a similar maximum transmission unit (MTU) size. However, when a datagram encounters networks with lower MTU limits, it must be divided, or fragmented, for smooth transmission across varied network types. This fragmentation process allows the datagram to adapt to diverse network requirements, ensuring reliable communication.

When a datagram is fragmented, each fragment is assigned its own header, replicating most of the original fields to maintain consistency and integrity. It's worth noting that a fragment can be further subdivided if it encounters a network with an even stricter MTU limit. This means a datagram might undergo multiple layers of fragmentation as it traverses various networks, adapting at each step to reach its destination.

In IPv4, fragmentation can be performed by either the source host or any routers along the route, though it's generally preferred to fragment only at the source. Reassembly of the fragments, however, is handled solely by the destination host. Since each fragment is treated as an independent datagram, fragments might travel different paths, but must all converge at the destination, making final reassembly there logical.

During fragmentation, essential header components are replicated in each fragment. The option field is an exception, as its inclusion depends on specific circumstances, discussed further in the next section. The device, whether a host or router, that fragments the datagram must update three critical fields—flags, fragmentation offset, and total length—to reflect the fragmented structure. All other fields are copied to maintain data integrity. Additionally, regardless of fragmentation, the checksum must be recalculated to uphold reliable data transmission.

## **2.2 Fields Related to Fragmentation**

- **Identification:**

The 16-bit identification field is essential for uniquely identifying a datagram originating from the source host. This identification number remains consistent and is duplicated across all fragments, exactly matching the original datagram. It acts as a crucial reference for the destination, facilitating the correct reassembly of the fragmented datagram. It is vital that all fragments with the same identification number are carefully grouped together, ensuring they are seamlessly reconstructed into a single, coherent datagram.

- **Flags:**



Figure 4: Flags used in fragmentation

This field consists of three bits, each with a unique function.

The first bit is reserved for future use or special control purposes, currently not in use. The second bit is the 'Do Not Fragment' bit, which is crucial for the datagram's transmission. If this bit is set to 1, it orders the system to keep the datagram intact and not to break it into smaller pieces. If the datagram cannot pass through a network without being fragmented, it is discarded, and an ICMP error message is sent back to the originating host to indicate the transmission failure. If this bit is set to 0, the datagram is allowed to be fragmented to fit the network's limitations. The third bit is the 'More Fragments' bit, which indicates whether the datagram is part of a sequence. A value of 1 means that the datagram is not the last in the sequence, while a 0 indicates that it is either the last fragment or the only fragment in the transmission. This sophisticated bit structure enables precise control over datagram delivery, preserving data integrity and allowing for necessary fragmentation, while keeping the receiver informed about the datagram's fragmentation status.

- **Fragmentation offset**

The 13-bit offset field is crucial for determining the position of a fragment within the entire datagram. It indicates the starting point of the fragment's data within the original datagram, with each unit representing an 8-byte block. An example is provided to illustrate this: a datagram that contains 4000 bytes of information has been divided into three separate fragments. This example, along with a diagram, helps to clarify how the offset field works to maintain the order and integrity of the fragments when they are part of a larger datagram. By using the offset field, the receiver can accurately reassemble the fragments back into the original datagram.

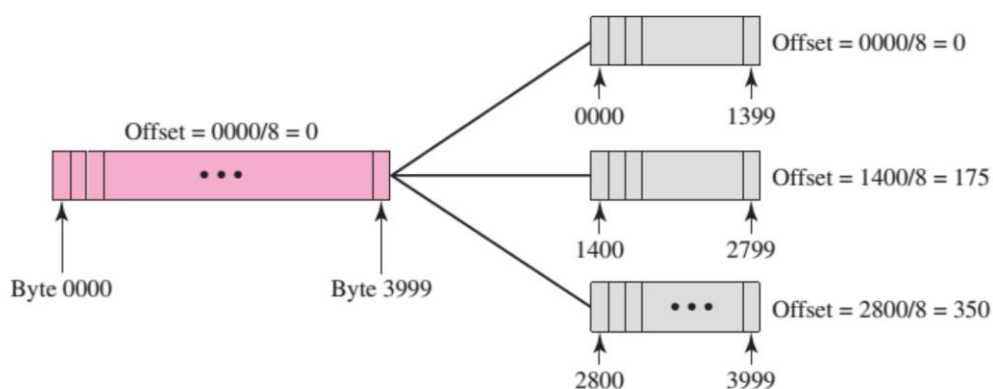


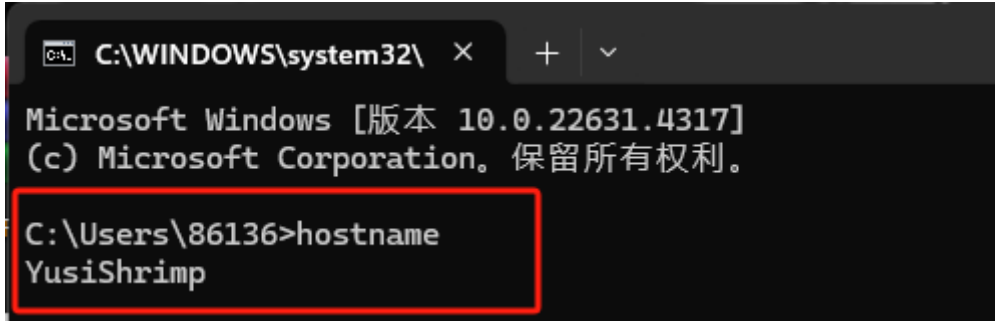
Figure 5: Fragmentation example

The original datagram's bytes are indexed from 0 to 3999. The first fragment contains bytes ranging from 0 to 1399, which corresponds to an offset of 0, calculated as  $0/8 = 0$ . The second fragment includes bytes from 1400 to 2799, with an offset of  $1400/8 = 175$ . The third fragment encompasses bytes 2800 to 3999, and its offset is  $2800/8 = 350$ .

The offset is determined in multiples of 8 bytes because the 13-bit offset field can only represent byte sequences up to 8191. This limitation requires that hosts or routers fragmenting the datagrams select a fragment size where the starting byte is a multiple of 8. This ensures that the offset values can be accurately represented within the constraints of the 13-bit field.

## 2. Lab Environment

I accomplish this lab at home using wired network with my laptop(Windows system).



```
C:\WINDOWS\system32\
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation。保留所有权利。

C:\Users\86136>hostname
YusiShrimp
```

Figure 6: Computer name

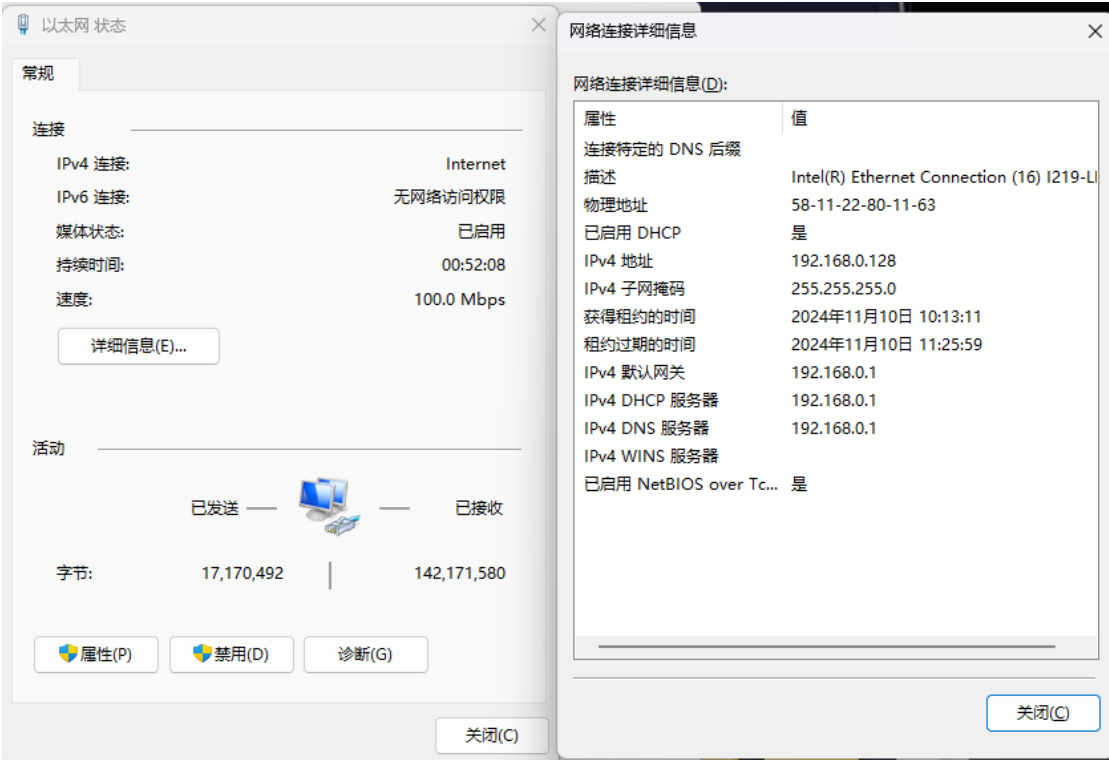


Figure 7: Detailed information about network

### 3. Experiment Steps and Results

#### 1) Task 1: Determine the Value of MTU

##### 1. Experiment steps

[1] Step 1: On Windows, execute the following command.

*“netsh interface ipv4 show subinterfaces”*

- You should get a list of all your network adapters installed on your PC. The MTU value is listed on the left.
- The effective MTU size may be smaller than the number displayed here, depending on your network and protocols used. For example, PPP connections (Point-to-Point Protocol) have a default MTU size of 1500 bytes and VPN connections have a default size of 1400.

[2] Step 2: To check the effectiveness of the MTU, say 1500 Bytes, type the following command to ping with an MTU size:

*“ping www.jnu.edu.cn -f -l 1472”*

- The “-f” marks packets that should not be fragmented in the ping, i.e., DF=1.
- The “-l”(lower case L) sets the size of ICMP data, i.e., **payload of ICMP message**. IP header has 20 Bytes without options, and ICMP message header has 8 Bytes. Thus, the effective ICMP payload size here is 1500-28 = 1472 Bytes.
- If your network has both IPv4 and IPv6 enabled, you can use “-4” to instruct ping to use IPv4.
- You can choose other web servers for ping commands in this lab.

[3] Step 3: If get successful replies, the current MTU is fine for connection. If receive error messages “Packet needs to be fragmented but DF set”, packets are to be fragmented but DF=1. Keep trying to ping with smaller size until get 4 successful replies.

##### 2. Results

```
C:\WINDOWS\system32\ x + v
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\86136>netsh interface ipv4 show subinterfaces

      MTU  MediaSenseState      输入字节      输出字节      接口
-----
4294967295      1              0          775312  Loopback Pseudo-Interface 1
1500          5              0              0  本地连接
1500          2  15922928448      7821625927  Wi-Fi
1500          1  19581660216  1173925298  以太网
1500          5              0              0  本地连接* 1
1500          5              0              0  本地连接* 2
1500          1          1968          58896  VMware Network Adapter VMnet1
1500          1              0          59721  VMware Network Adapter VMnet8

C:\Users\86136>
```

Refer to “以太网” cuz I'm using wired network to finish this lab

Figure 8: MTU size of every network in my laptop

```
C:\WINDOWS\system32\
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation。保留所有权利。

C:\Users\86136>ping www.jnu.edu.cn -f -l 1472
正在 Ping wafmnc.jnu.edu.cn.ctdns.cn [120.240.113.2] 具有 1472 字节的数据:
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。

120.240.113.2 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 0, 丢失 = 4 (100% 丢失),

C:\Users\86136>ping www.jnu.edu.cn -f -l 1480
正在 Ping wafmnc.jnu.edu.cn.ctdns.cn [120.240.113.2] 具有 1480 字节的数据:
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。

120.240.113.2 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 0, 丢失 = 4 (100% 丢失),

C:\Users\86136>
```

1472+28(IP header's length) ≤ 1500(MTU), So this packet can be sent without fragmentation

1480+28(IP header's length) > 1500(MTU), So it can not be sent without fragmentation

Figure 9: Check MTU on Windows

## 2) Task 2: Modify the Value of MTU

### 1. Experiment steps:

[1] Step 1: On Windows, execute the following command.

- “netsh interface ipv4 set subinterface “Local Area Connection” mtu=1490 store=persistent”
- Change the interface name to whatever you’re using, e.g., “Local Area Connection 2” or “本地连接 1”. Capture a Trace of IP Fragmentation
  - You need to add 28 Bytes on to the value you were using in your pings.

[2] Step 2: Try Step 2 and Step 3 in Task 1 to verify the value of new MTU.

[3] Step 3: Restore the MTU value after the lab.

### 2. Results:

```
管理员: C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation. 保留所有权利。

C:\Windows\System32>netsh interface ipv4 set subinterface "以太网" mtu=1490 store=persistent
确定。

C:\Windows\System32>netsh interface ipv4 show subinterfaces
```

MTU	MediaSenseState	输入字节	输出字节	接口
4294967295	1	0	775312	Loopback Pseudo-Interface 1
1500	5	0	0	本地连接
1500	2	15922928448	7821625927	WI AM
1490	1	19634999556	1179817261	以太网
1500	5	0	0	本地连接* 1
1500	5	0	0	本地连接* 2
1500	1	2296	75680	VMware Network Adapter VMnet1
1500	1	0	76649	VMware Network Adapter VMnet8

MTU has been changed successfully

Figure 10: Modifying MTU on Windows

```
C:\WINDOWS\system32\
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\86136>ping www.jnu.edu.cn -f -l 1472
正在 Ping wafmnc.jnu.edu.cn.ctdns.cn [120.240.113.2] 具有 1472 字节的数据:
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。
需要拆分数据包但是设置 DF。

120.240.113.2 的 Ping 统计信息:
数据包: 已发送 = 4, 已接收 = 0, 丢失 = 4 (100% 丢失),

C:\Users\86136>ping www.jnu.edu.cn -f -l 1462
正在 Ping wafmnc.jnu.edu.cn.ctdns.cn [120.240.113.2] 具有 1462 字节的数据:
来自 120.240.113.2 的回复: 字节=1462 时间=18ms TTL=54
来自 120.240.113.2 的回复: 字节=1462 时间=20ms TTL=54
来自 120.240.113.2 的回复: 字节=1462 时间=17ms TTL=54
来自 120.240.113.2 的回复: 字节=1462 时间=17ms TTL=54

120.240.113.2 的 Ping 统计信息:
数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 17ms, 最长 = 20ms, 平均 = 18ms

C:\Users\86136>
```

MTU now is 1490 and then  $1372+28 > 1490$

$1462+28 \leq 1490$

Figure 11: Check the value of MTU



```

C:\Windows\System32>netsh interface ipv4 set subinterface "以太网" mtu=1500 store=persistent
确定。

C:\Windows\System32>netsh interface ipv4 show subinterfaces

    MTU  MediaSenseState      输入字节      输出字节      接口
-----
4294967295      1              0          775312  Loopback Pseudo-Interface 1
1500          5              0              0  本地连接* 1
1500          5              0              0  本地连接* 2
1500          2  15922928448  7821625927  WLAN
1500          1  19647297519  1180698047  以太网
1500          5              0              0  本地连接* 1
1500          5              0              0  本地连接* 2
1500          1          2624          78598  VMware Network Adapter VMnet1
1500          1              0          79472  VMware Network Adapter VMnet8

C:\Windows\System32>_
  
```

Figure 12: Restore the value of MTU

### 3) Task 3: Capture a Trace of IP Fragmentation

#### 1. Experiment steps:

- [1] Step 1: Suppose the MTU is 1500. Launch Wireshark and start a capture with a filter of "ip.proto==1".
- [2] Step 2: Execute the following command:

*"ping www.jnu.edu.cn -l 3000 -n 1"*

After the ping command is complete, return to Wireshark and stop the trace.

- The "-n" specifies the number of ICMP Echo Request messages sent. The default is 4. "-n 1" will generate one ICMP Echo Request.
- By default, Wireshark will try to reassemble fragments. You can disable it in preferences setting of IPv4 (Edit→Preferences→Protocols→IPv4→Uncheck "Reassemble fragmented IPv4 datagrams"), or in the right-click menu of an IP packets from captured result.

- [3] Step 3: Locate all fragments of the ICMP Echo Request, and fill the following table:

Fields	Fragmentation 1	Fragmentation 2	Fragmentation 3
"Identification"	0xcf78(53112)	0xcf78(53112)	0xcf78(53112)
"MF"	1	1	0
"DF"	0	0	0
"Fragment Offset" (decimal value)	0	1480	2960
Size of transmitted data (IP payload)	1472Bytes	1480Bytes	48Bytes

#### 2. Results

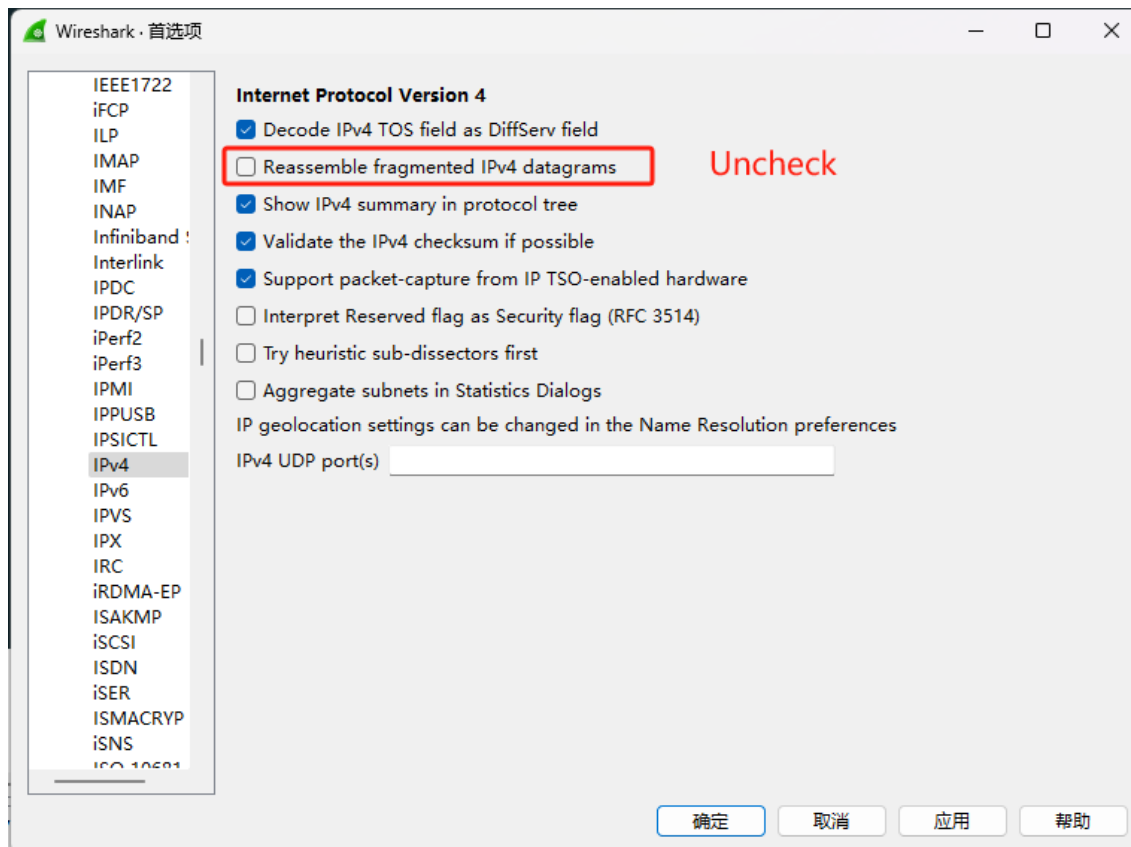


Figure 13: Modify the related settings of wireshark

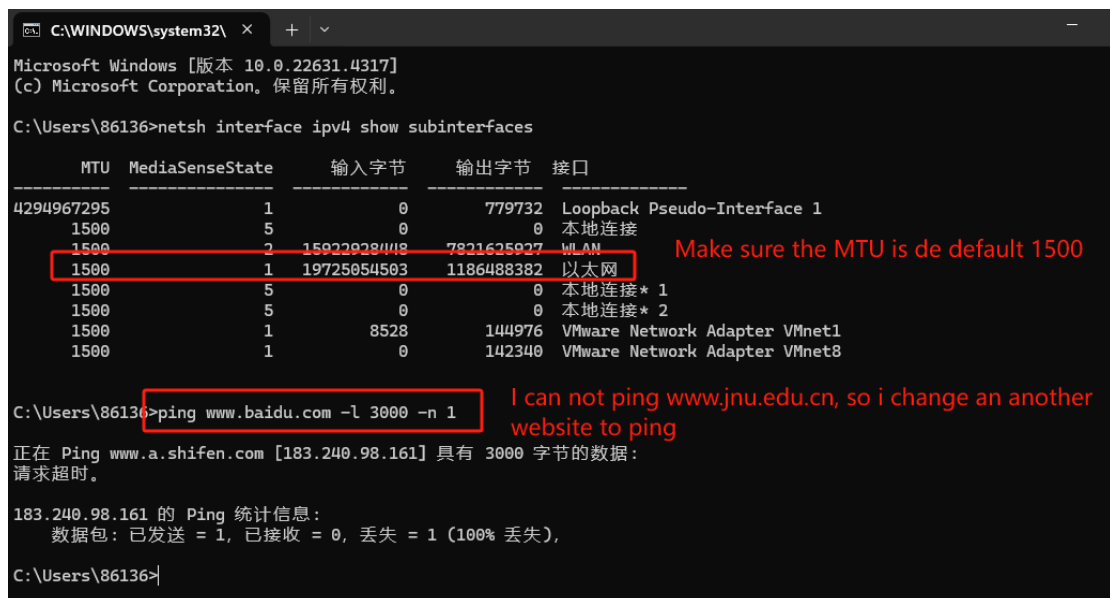
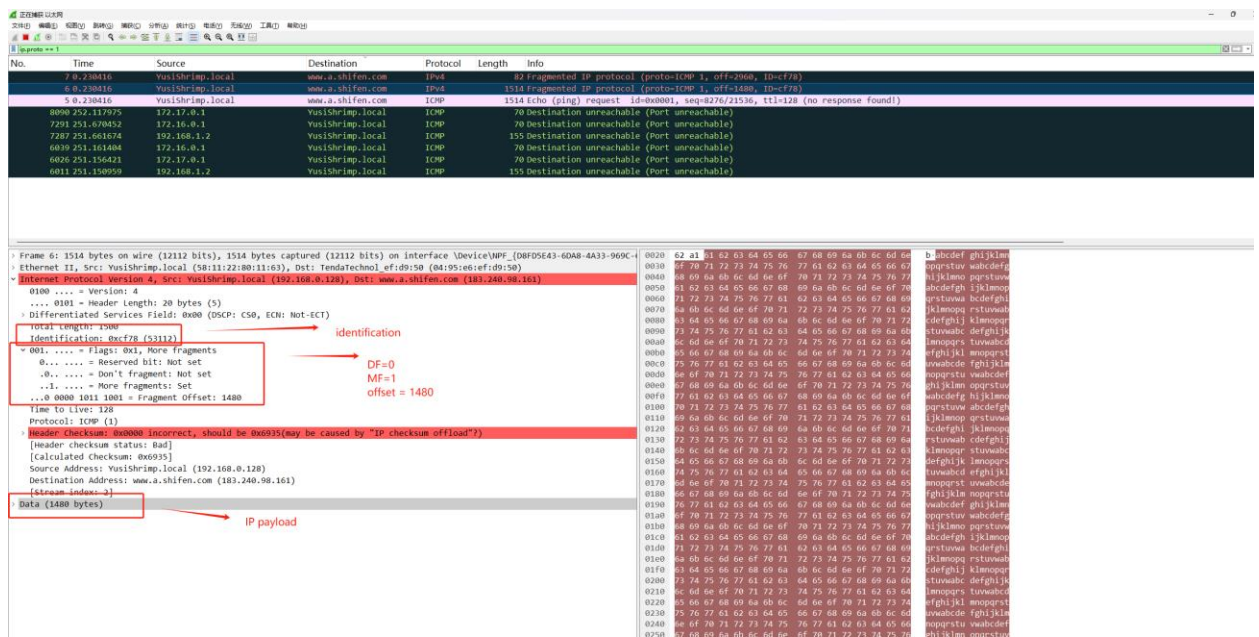
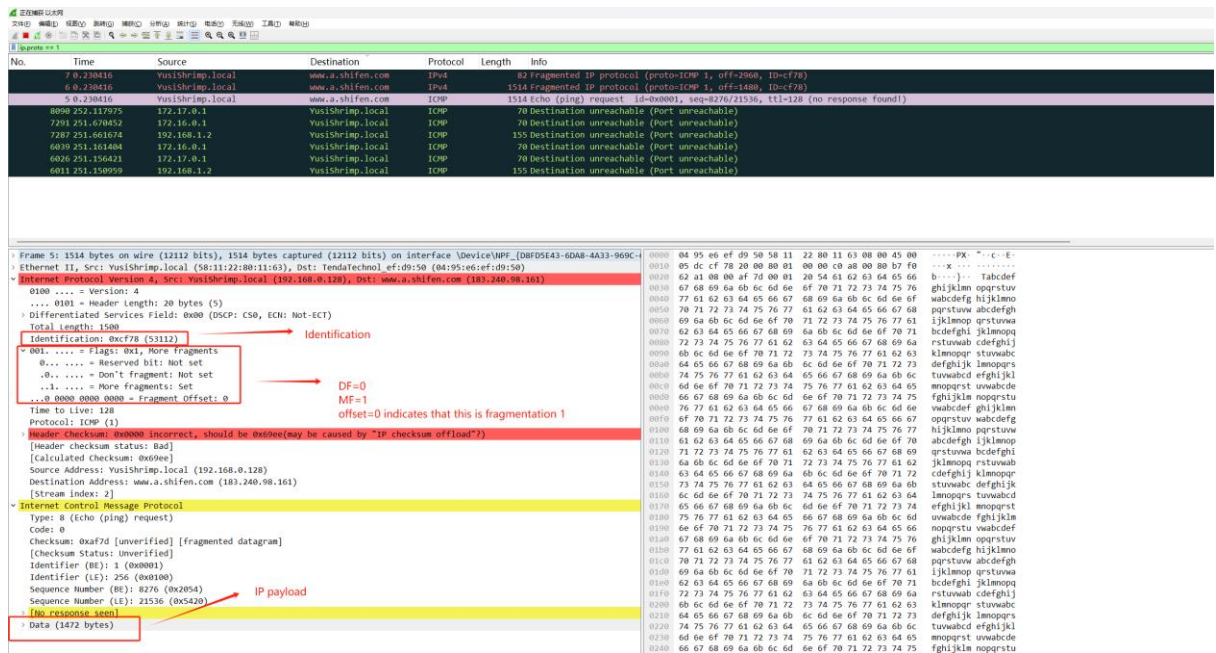


Figure 14: Ping command in CMD windows



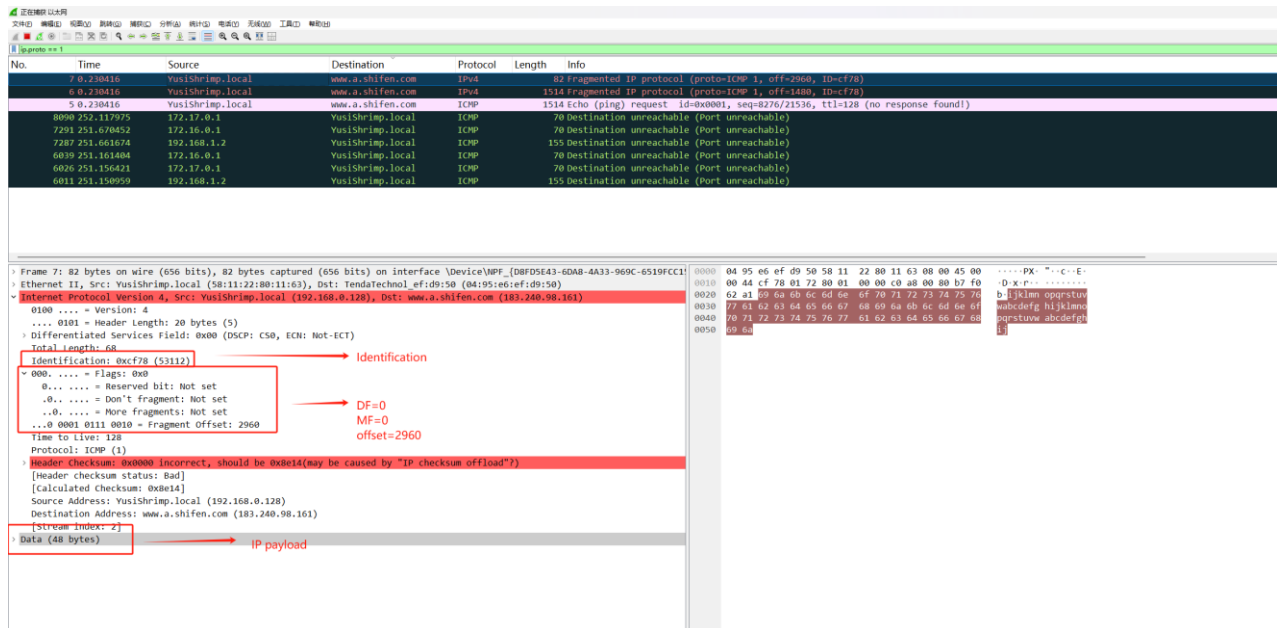


Figure 17: Fragmentation 3

#### 4) Task 4: IP fragmentation & reassembly (Optional, but strongly suggested!!!)

##### 1. Experiment steps:

##### [1] Step 1: Host A ping Host B to ensure the reachability of ping between A and B.

- The ping maybe blocked between Host A and B when they are connected to different networks. This is because some networks or computers may use strict network configurations.
- In this case, I suggest you to use computers in the lab.

##### [2] Step 2: Both Host A and B start Wireshark to capture the IP packets exchanged between A and B.

- You can setup a filter using the IP address of computer on the other end. For example, on Host A: ip.addr==IP\_B; on Host B: ip.addr==IP\_A.
- You can also use filter of ip.dst or ip.src to filter out packet on only one communication direction.

##### [3] Step 3: Host A ping Host B using a large ICMP message, using the similar operations as Task 3.

##### [4] Step 4: Investigate the captured trace on both Host A and Host B.

##### [5] Step 5: Describe the process of fragmentation and reassembly for an IP packet sent from Host A to Host B.

##### 2. Results:

Before this task begin, the two hosts are connected to the same wireless network



Figure 18: Host A's network information



Figure 19: Host B's network information

Two computers can use the ping command to issue ICMP requests and replies to each other. The following figure is the result of Host A issue a ping command to Host B:

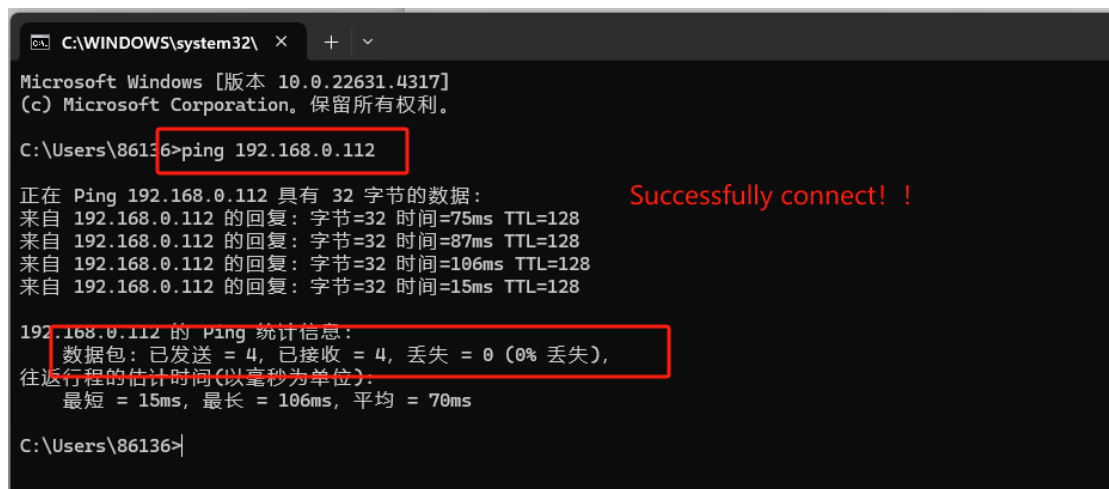


Figure 20: Connect Host A and B successfully

At the same time, Host A can use Wireshark to capture traces of this communication. Here I set Host A's Wireshark to filter traces that contain Host B's IP address:

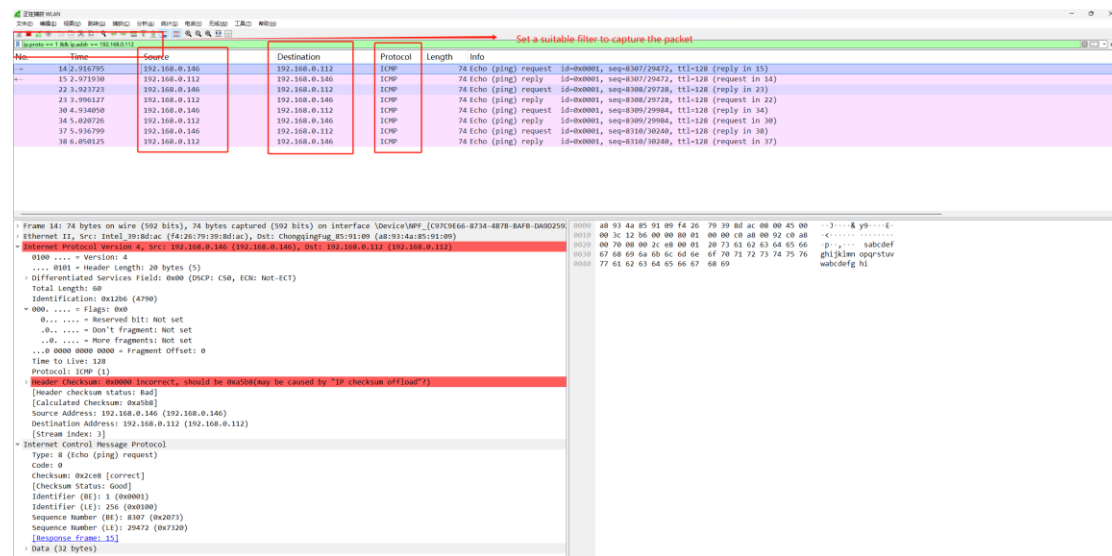


Figure 21: Trace captured by Host A

Host B can also capture traces of this communication. I set Host B's Wireshark to filter traces that contain Host A's IP address:

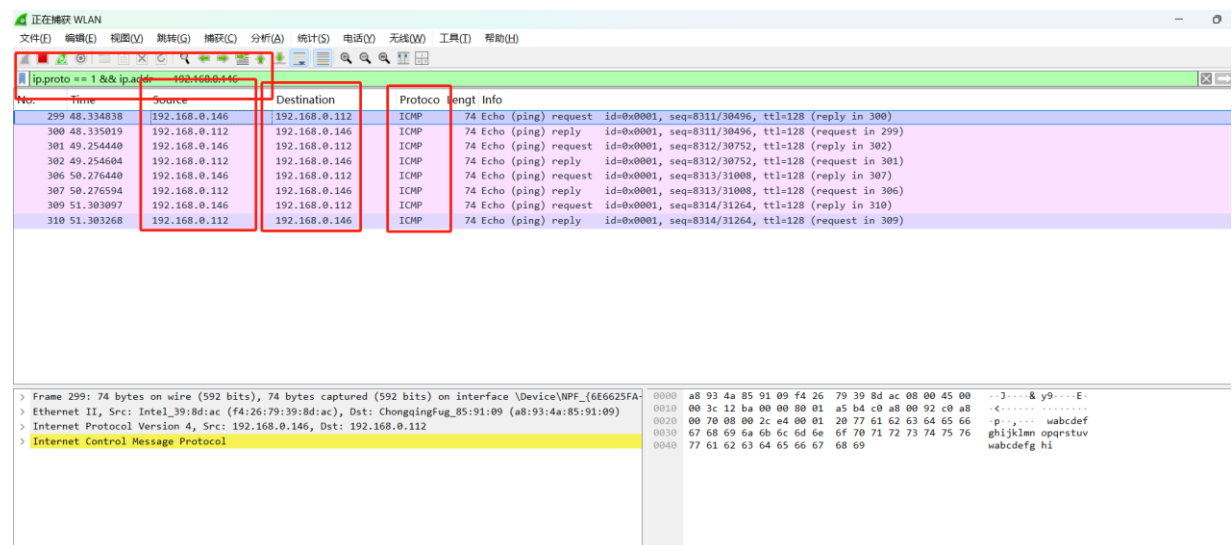


Figure 22: Trace captured by Host B

Next, check the MTU sizes of the respective network interfaces of the two computers. They are both standard 1500 Bytes:



```

C:\WINDOWS\system32\ x + v
C:\Users\86136>netsh interface ipv4 show subinterfaces
Host A

```

MTU	MediaSenseState	输入字节	输出字节	接口
4294967295	1	0	801707	Loopback Pseudo-Interface 1
1500	5	0	0	本地连接
1500	1	5351214	2498291	WLAN
1500	2	20896771042	1382770416	以太网
1500	5	0	0	本地连接* 1
1500	5	0	0	本地连接* 2
1500	1	11480	205820	VMware Network Adapter VMnet1
1500	1	0	201856	VMware Network Adapter VMnet8

```

C:\Users\86136>

```

Figure 23: Host A's MTU

```

C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation. 保留所有权利。
Host B
C:\Users\hp>netsh interface ipv4 show subinterfaces

```

MTU	MediaSenseState	输入字节	输出字节	接口
4294967295	1	0	393860	Loopback Pseudo-Interface 1
1500	1	3282047503	153436070	WLAN
1500	5	0	0	本地连接* 1
1500	5	0	0	本地连接* 2

```

C:\Users\hp>

```

Figure 24: Host B's MTU

Next, Host A ping Host B using a large ICMP message, using the similar operations as Task 3:

```

C:\WINDOWS\system32\ x + v
C:\Users\86136>netsh interface ipv4 show subinterfaces

```

MTU	MediaSenseState	输入字节	输出字节	接口
4294967295	1	0	801707	Loopback Pseudo-Interface 1
1500	5	0	0	本地连接
1500	1	5351214	2498291	WLAN
1500	2	20896771042	1382770416	以太网
1500	5	0	0	本地连接* 1
1500	5	0	0	本地连接* 2
1500	1	11480	205820	VMware Network Adapter VMnet1
1500	1	0	201856	VMware Network Adapter VMnet8

```

C:\Users\86136>ping 192.168.0.112 -l 3000 -n 1
正在 Ping 192.168.0.112 具有 3000 字节的数据:
来自 192.168.0.112 的回复: 字节=3000 时间=31ms TTL=128

192.168.0.112 的 Ping 统计信息:
    数据包: 已发送 = 1, 已接收 = 1, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 31ms, 最长 = 31ms, 平均 = 31ms

```

Figure 25: Host A pings Host B using a large ICMP message(>=MTU)



Similarly, I turned off Wireshark's reassembly IPv4 datagrams option on both Host A and Host B:

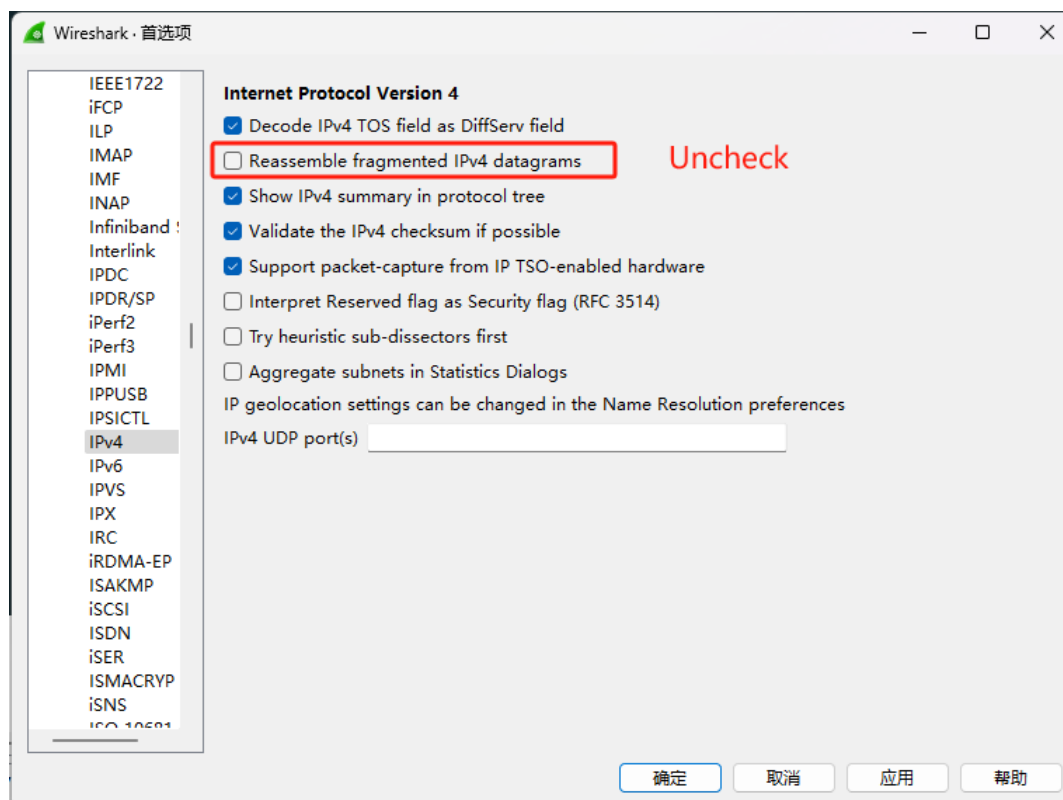


Figure 26: Uncheck the setting on both Host A and B

On Host A, the captured trace is as follows:

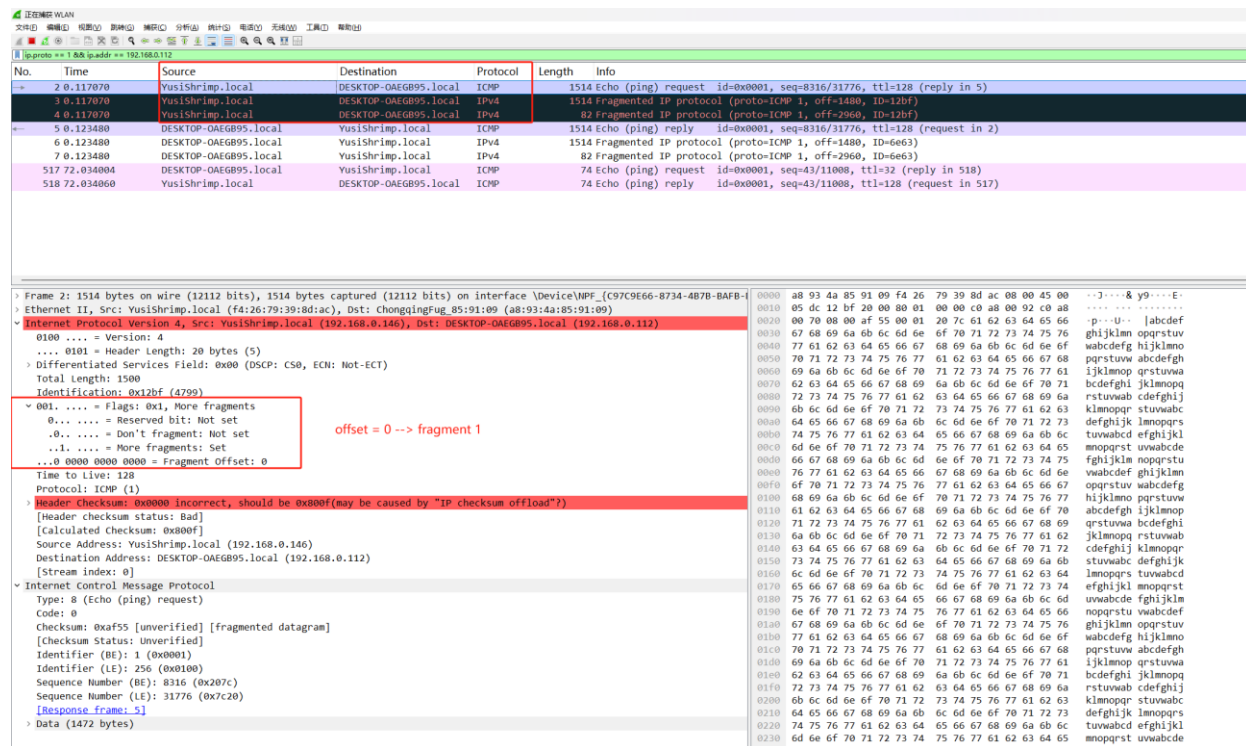


Figure 27: Host A's ICMP request fragment 1

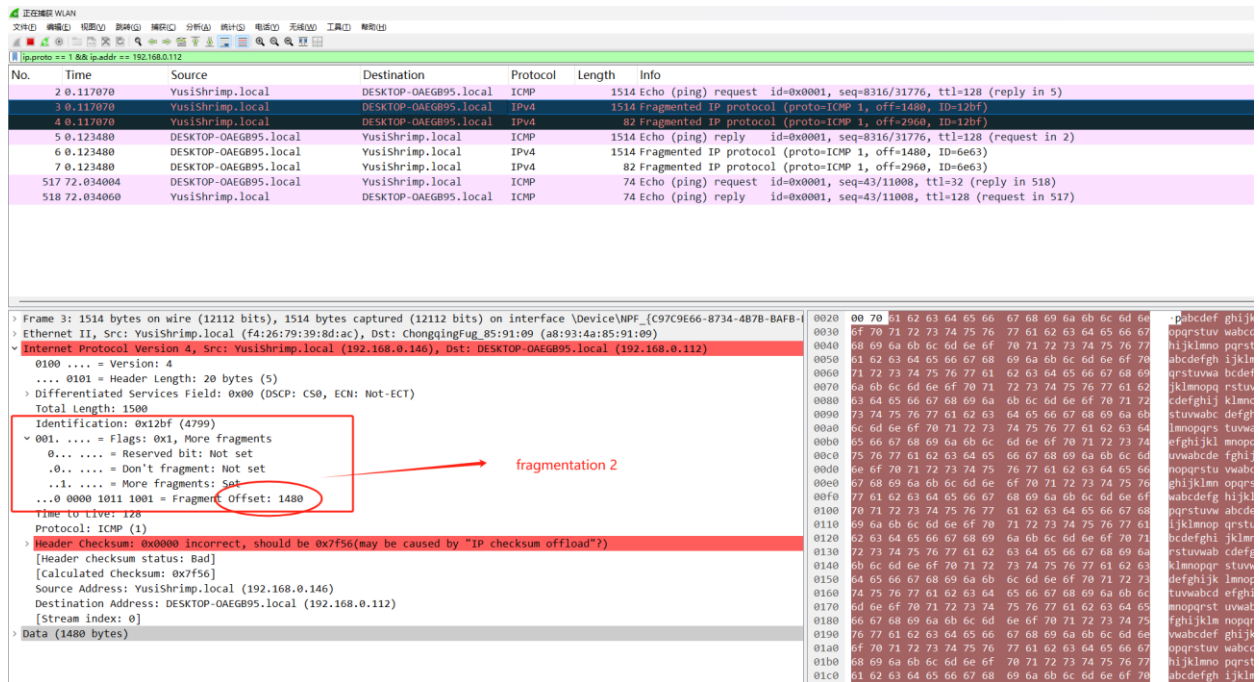


Figure 28: Host A's ICMP request fragmentation 2

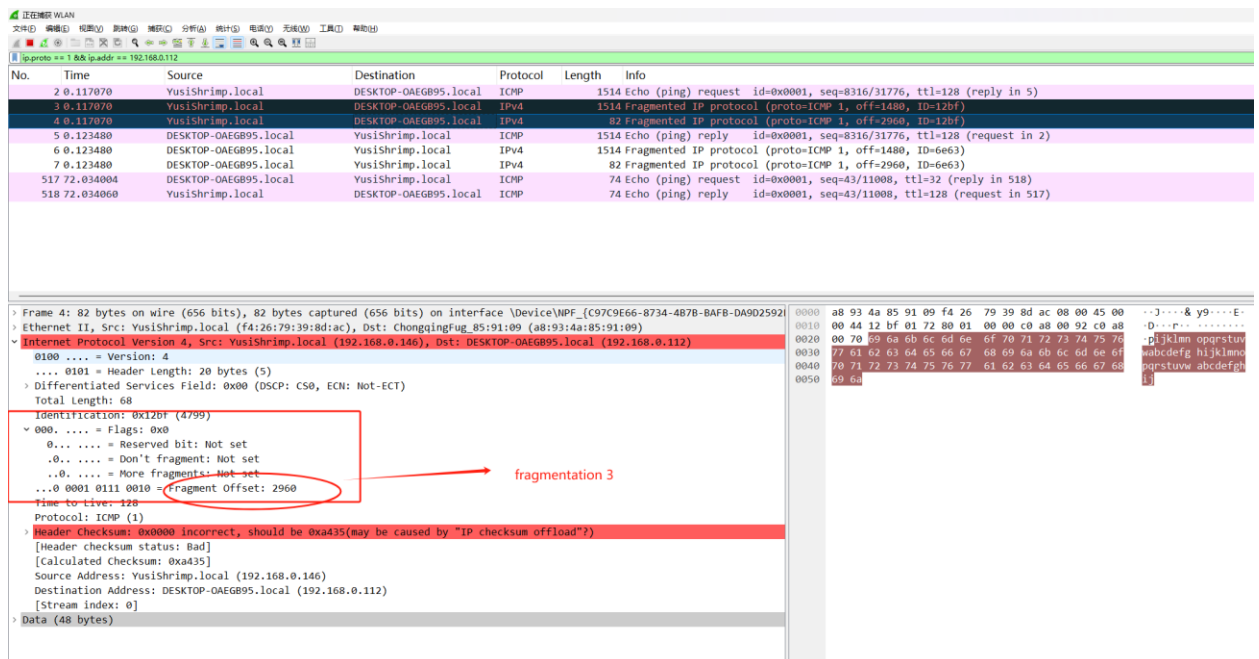
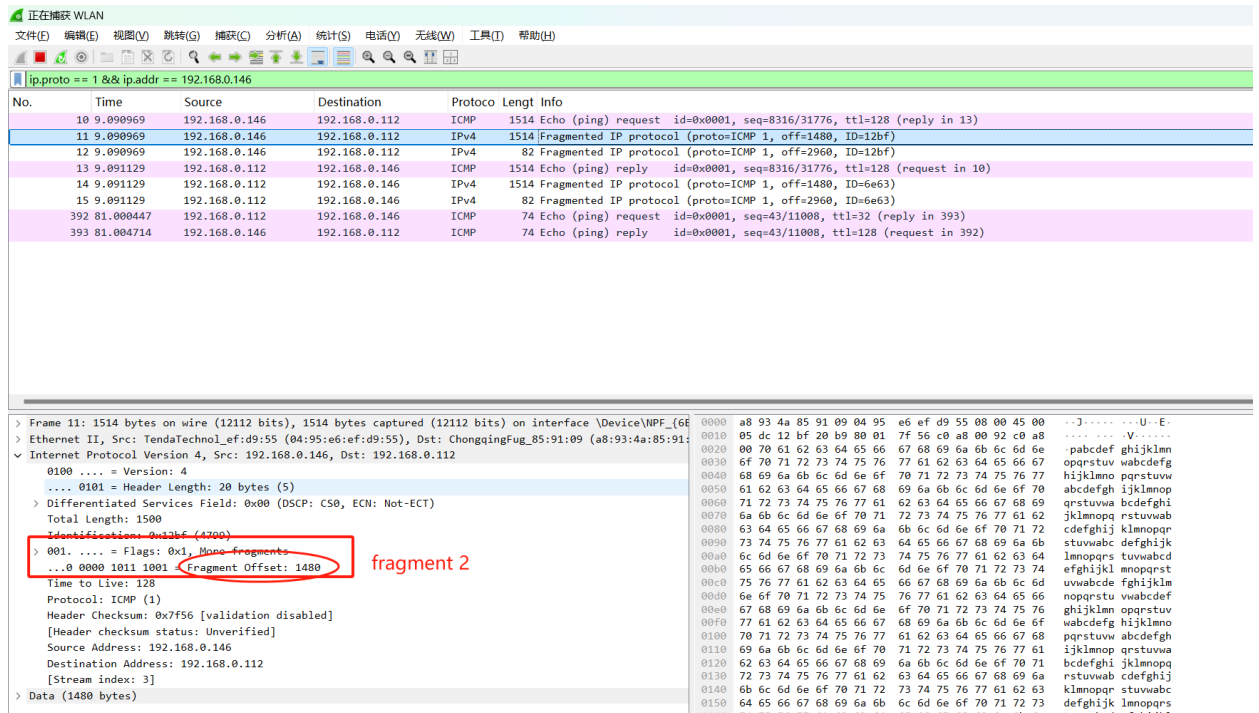
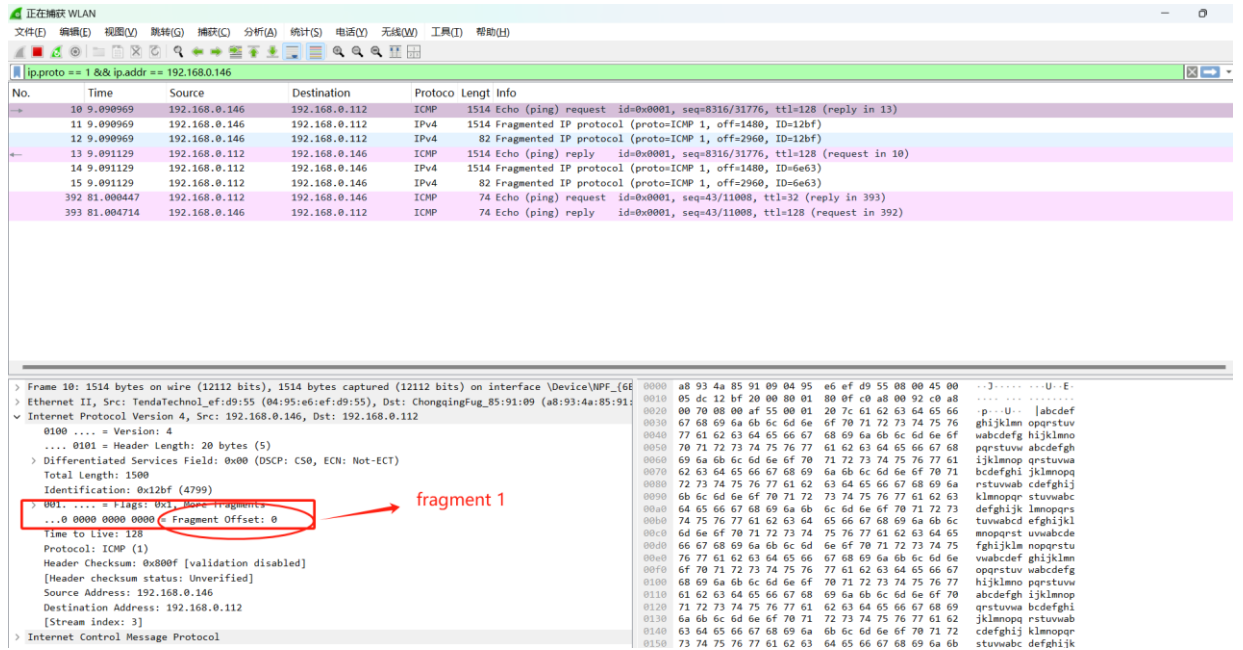


Figure 29: Host A's ICMP request fragmentation 3

On Host B, the captured trace is as follows:



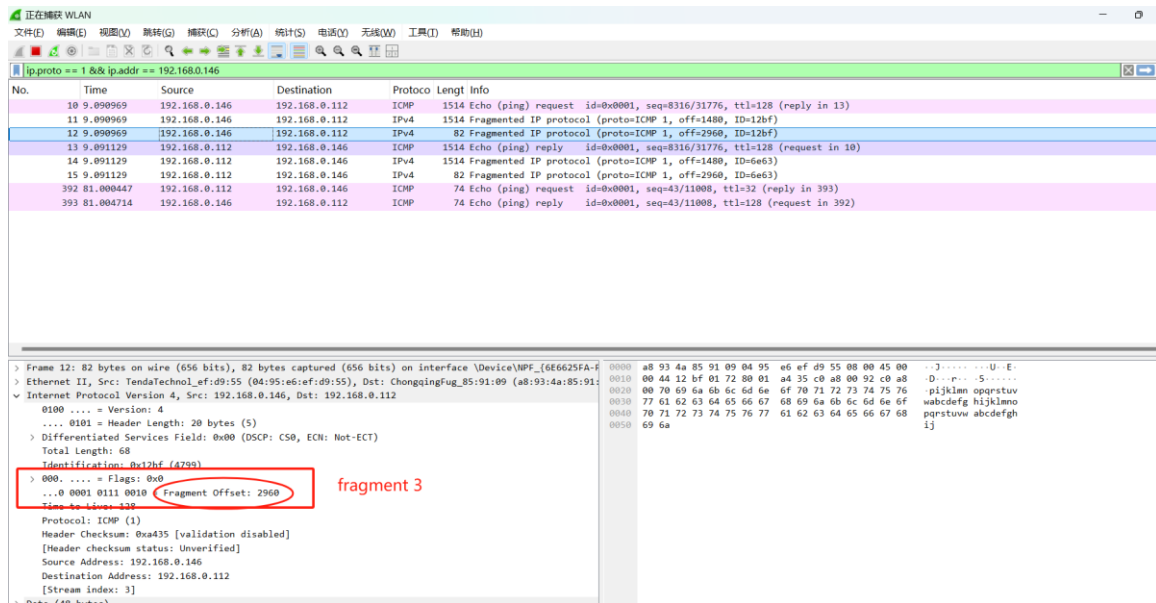


Figure 32: Host B's ICMP reply fragment 3

The process of fragmentation and then reassembling an IP packet that travels from Host A to Host B can be described as follows: When Host A sends a 3000-byte ICMP data packet to Host B, and this packet exceeds the network's Maximum Transmission Unit (MTU) of 1500 bytes, fragmentation is necessary.

Host A splits the large packet into smaller segments, each equipped with its own IP header. These headers share a common identification number, a More Fragments (MF) flag, and a fragment offset value that indicates the segment's position relative to the original packet. These segments are then transmitted separately across the network towards Host B.

Upon receipt, Host B utilizes the identification number to gather all the segments that belong to the initial packet and employs the fragment offsets to arrange them in the correct sequence. The MF flag assists Host B in determining if additional segments are anticipated. Once all segments have been received, Host B reconstructs the original packet for further processing.

Subsequently, for the response, Host B must also fragment its reply and dispatch these fragments to Host A. The process of fragmentation and reassembly for this return ICMP message mirrors the initial request from Host A to Host B.

## [4] Others

### 1) Answering the questions"

**Q1: Most often IP packets in normal operation are not fragmented. But the receiver must have a way to be sure. How does the receiver computer determine whether a packet is fragmented or not?**

**A1: The status of IP packet fragmentation is determined by two specific fields within the IP header:**

**the "More Fragments" (MF) flag and the "Fragment Offset" field.**

- **More Fragments Flag:** The MF flag, when set (value of 1), signifies that the current packet is part of a fragmented sequence and that subsequent fragments are forthcoming. If the flag is not set, it suggests that the packet is either the final fragment or that the packet has not been fragmented at all.
- **Fragment Offset Field:** This field denotes the starting position of the fragment's data in relation to the beginning of the data in the original, unfragmented packet. The measurement is in units of 8-byte blocks. A non-zero Fragment Offset value indicates that the packet is indeed a fragment.

By examining these two fields together, one can ascertain the fragmentation status of an IP packet: if either the MF flag is set or the Fragment Offset is non-zero, the packet is identified as a fragment. Conversely, if the MF flag is unset and the Fragment Offset is zero, the packet is considered unfragmented.

**Q2: If an IP packet is fragmented, how does the receiver identify all fragments?**

**A2:** When an IP packet undergoes fragmentation, each resulting piece is considered a separate IP packet, complete with its own header. The destination system utilizes the data within these headers to recognize which fragments are part of the same original packet and to reassemble them in the correct sequence.

Each piece of the fragmented packet includes an identification field within its IP header. This field contains a value that is unique to the original packet and consistent across all of its fragments. The destination system employs this identification field to correlate fragments that pertain to the same packet.

Fragments can arrive at the destination in any order. To accurately reassemble them, the destination system relies on the fragment offset field in the IP header, which specifies the position of the fragment's data within the original packet. The system then aligns the data from each fragment according to this offset.

The destination system also needs to determine when it has received all fragments of the original packet. This is signaled by the More Fragments (MF) flag in the IP header. If the MF flag is set, it indicates that additional fragments are anticipated. If the MF flag is not set, it signifies that the current fragment is the final one. Once the destination system has received the last fragment and all preceding fragments (as deduced from the offsets and data lengths), it can reconstruct the complete original packet.

**Q3: What is the meaning of filter "ip.proto==1"? Can you use filter "icmp" to obtain all fragments in Task 3 of this lab? What is the difference between the display results of these two filters (i.e., "ip.proto==1" and "icmp") in Wireshark? Explain the reason.**

**A3:** When an IP packet undergoes fragmentation, each resulting piece is considered a separate IP packet, complete with its own header. The destination system utilizes the data within these headers to recognize which fragments are part of the same original

packet and to reassemble them in the correct sequence.

Each piece of the fragmented packet includes an identification field within its IP header. This field contains a value that is unique to the original packet and consistent across all its fragments. The destination system employs this identification field to correlate fragments that pertain to the same packet.

Fragments can arrive at the destination in any order. To accurately reassemble them, the destination system relies on the fragment offset field in the IP header, which specifies the position of the fragment's data within the original packet. The system then aligns the data from each fragment according to this offset.

The destination system also needs to determine when it has received all fragments of the original packet. This is signaled by the More Fragments (MF) flag in the IP header. If the MF flag is set, it indicates that additional fragments are anticipated. If the MF flag is not set, it signifies that the current fragment is the final one. Once the destination system has received the last fragment and all preceding fragments (as deduced from the offsets and data lengths), it can reconstruct the complete original packet.

**Q4: What are the advantages for large MTU and small MTU respectively? Tips: see wiki for more details about MTU.**

**A4: Regarding large MTUs:**

Large MTUs enable the transmission of more data per packet, which decreases the relative overhead of headers and enhances the efficiency of data transmission. This can result in higher throughput, as the network can deliver a greater amount of payload data in proportion to the total data transmitted, including headers. Fewer packets need to be processed for a given amount of data, which also lightens the load on network devices. However, larger packets may increase latency due to the longer time required for their transmission. Moreover, if packet loss occurs, the retransmission of these larger packets can consume more bandwidth and time, possibly causing additional delays.

**Regarding small MTUs:**

Small MTUs can minimize latency, which is beneficial for real-time applications such as voice or video communications where the timeliness of data delivery is more important than maximizing throughput. Smaller packets are also less likely to face issues with network devices that do not support larger MTUs, ensuring better compatibility and reducing the chance of fragmentation. In environments with frequent packet loss, smaller MTUs can facilitate quicker recovery since less data is needed to be retransmitted for each lost packet. However, the downside is that the overhead of packet headers becomes more significant, which can lead to reduced efficiency and potentially lower throughput.

2) Some personal thought about this lab

By conducting this experiment, I gained a deeper understanding of the maximum transmission unit (MTU) and its role in network communications. I was able to

witness the processes of data packet fragmentation and reassembly during network transmission by adjusting the size of the data packets and the MTU settings on the network interface. This hands-on experience has significantly enhanced my comprehension of the intricacies of computer networking.