aselsan

Internship Project Analysis Report
(Wireshark Analysis Included)

ICMP Packet Watcher

Directorate:

Defense System Technologies
Group Presidency (SST)

Unit:

Embedded and Real-Time
Software Design

Name and Surname:

Yekta Güngör PARLAK

September 2022
ANKARA

Table of Contents

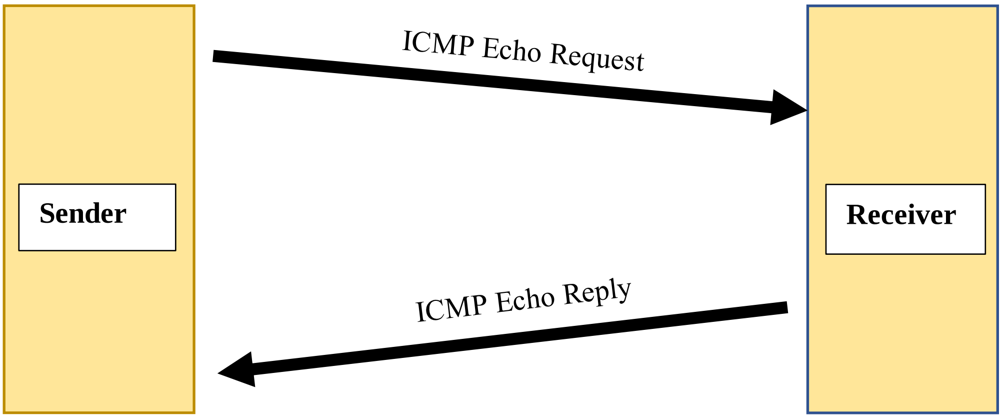
CHAPTER 1: INTRODUCTION

1.1 Protocols: What Are They?

So, a protocol is basically a way of ensuring that devices are able to talk to each other effectively. In most cases, an individual protocol describes how communication is accomplished between one particular software or hardware element in two or more devices. In the context of the OSI Reference Model, a protocol is formally defined as a set of rules governing communication between entities at the same Reference Model layer.

1.2 ICMP Definition

The Internet Protocol (IP) is the key network layer protocol that implements the TCP/IP protocol suite. One of the provisions it lacks is some way to allow errors to be reported back to a transmitting device, and for tests and special tasks to be accomplished. However, these capabilities are necessary for the operation of an internetwork, so TCP/IP defines an protocol for IP that provides them. ICMP is assigned Protocol Number 1 in the IP suite.



ICMP originally created to allow the reporting of set of error conditions, ICMP messages are now used to implement a wide range of error-reporting, feedback and testing capabilities. While each message type is unique, they are all implemented using a common message format, sent and received based on relatively simple protocol rules. This makes ICMP one of the easiest TCP/IP protocols to understand.

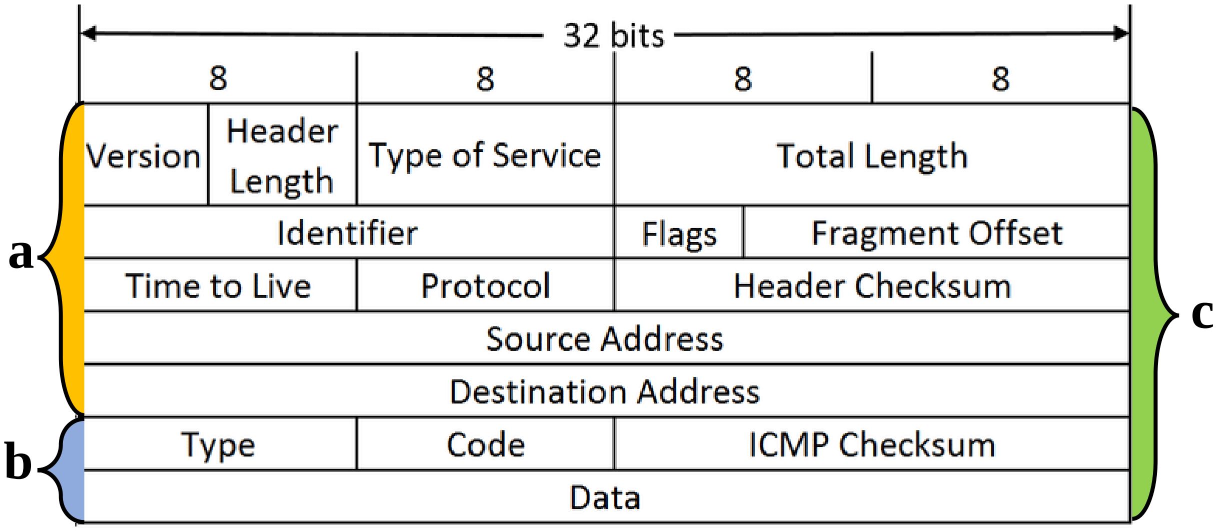
RFC 792 was the initial defining standard for ICMP, titled as simply Internet Control Message Protocol. It was published at the same time as the standard for IP, which was RFC 791.

The IP protocol alone provides no direct way to do the following:

- For an end system to learn the fate of IP packets that fail to make it to their destinations.
- For obtaining diagnostic information (e.g., which routers are used along a path or a method to estimate the round-trip time).

To address these deficiencies, a special protocol called the Internet Control Message Protocol (ICMP) is used in conjunction with IP to provide diagnostics and control information related to the configuration of the IP protocol layer and the disposition of IP packets.

The ICMP packet is encapsulated in an IPv4 packet. The general composition of the IP datagram is as follows:



(**a:** IP Header, **b:** ICMP Header, **c:** IP Packet)

ICMP Common Message Format		
Field Name	**Size (Bytes)**	**Description**
Type	1	Identifies the ICMP message type. For ICMPv6, values from 0 to 127 are error messages and values 128 to 255 are informational messages. Common values for this field are given in the table in the topic on ICMP message classes and types.
Code	1	Identifies the "subtype" of message within each ICMP message Type value. Thus, up to 256 "subtypes" can be defined for each message type. Values for this field are shown in the individual ICMP message type topics.
Checksum	2	16-bit checksum field that is calculated in a manner similar to the IP header checksum in IPv4. It provides error detection coverage for the entire ICMP message.
Message Body / Data	32 (default)	Contains the specific fields used to implement each message type. This is the unique part of the message.

The message body typically contains one or several fields that carry information. In ICMP messages, all ICMP error messages include a portion of the original IP datagram that led to the ICMP error message. This helps in diagnosing the problem that caused the ICMP message to be generated, by allowing the error to be communicated to higher layers.

1.3 What is Ping?

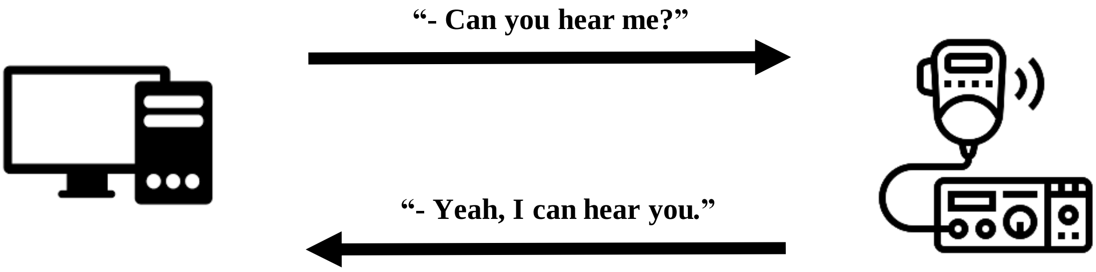
ICMP and ping are two different things although they are related. ICMP is a protocol that controls how messages are sent between devices. The echo requests and replies the ICMP protocol sends are commonly referred to as pings. So while a ping is produced using ICMP, it is not ICMP.

Ping is actually an acronym for the words 'Packet Internet Groper'. Another is that it is in fact not an acronym at all, but a noun that was adopted from a verb that the US Navy that uses to describe what its submarines do when looking for objects under the sea.

Their submarines send out sonar waves and then wait for a return wave when it bounces off something, such as another sub, whale, ocean floor etc. This, in turn, was adopted from bats and dolphins, who navigate in roughly the same way. This is what a system administrator does when Ping is used. As such, Ping has also evolved into a verb in the computer industry, and it is used in somewhat the same manner of the Navy.

Ping sends out IP packets with ICMP Echo Request to the destination and waits for its reply (IP packet with ICMP Echo Reply).

The Ping utility is essentially a system administrator's tool that is used to see if a computer is operating and also to see if network connections are intact. Ping uses the Internet Control Message Protocol (ICMP) Echo function which is detailed in RFC 792. A small packet is sent through the network to a particular IP address. This packet contains 32 data bytes and 8 bytes of ICMP protocol reader information. The computer that sent the packet then waits (or 'listens') for a return packet. If the connections are good and the target computer is up, a good return packet will be received.



1.4 Implementation of ICMP

In RFC 1122: Requirements for Internet Host, Section 3.2.2.6, says "Every host must implement an ICMP Echo server function that receives Echo Requests and sends corresponding Echo Replies." To send ICMP, a device must have Layer-3/4 support and if a device getting Ipv4 address, then it is possible to respond a ICMP echo respond from it. If a device is capable of running IPv4, then it will reply the ICMP request ping.

However if a firewall is implementing afterwards, the echo request reply would not work under this circumstance.

1.5 OSI Model

Open system interconnection (OSI) reference model consists of seven layers or seven steps which concludes the overall communication system.

It is important to understand this OSI model as each of the software applications works based on one of the layers in this model.

Architecture Of The OSI Reference Model:

Group	#	Layer Name	Key Responsibilities	Data Type Handled	Scope	Common Protocols and Technologies
Lower Layers	1	**Physical**	Encoding and Signaling; Physical Data Transmission; Hardware Specifications; Topology and Design	Bits	Electrical or light signals sent between local devices	(Physical layers of most of the technologies listed for the data link layer)
	2	**Data Link**	Logical Link Control; Media Access Control; Data Framing; Addressing; Error Detection and Handling; Defining Requirements of Physical Layer	Frames	Low-level data messages between local devices	IEEE 802.2 LLC, Ethernet Family; Token Ring; FDDI and CDDI; IEEE 802.11 (WLAN, Wi-Fi); HomePNA; HomeRF; ATM; SLIP and PPP
	3	**Network**	Logical Addressing; Routing; Datagram Encapsulation; Fragmentation and Reassembly; Error Handling and Diagnostics	Datagrams / Packets	Messages between local or remote devices	IP; IPv6; IP NAT; IPsec; Mobile IP; ICMP; IPX; DLC; PLP; Routing protocols such as RIP and BGP
	4	**Transport**	Process-Level Addressing; Multiplexing/Demultiplexing; Connections; Segmentation and Reassembly; Acknowledgments and Retransmissions; Flow Control	Datagrams / Segments	Communication between software processes	TCP and UDP; SPX; NetBEUI/NBF
Upper Layers	5	**Session**	Session Establishment, Management and Termination	Sessions	Sessions between local or remote devices	NetBIOS, Sockets, Named Pipes, RPC
	6	**Presentation**	Data Translation; Compression and Encryption	Encoded User Data	Application data representations	SSL; Shells and Redirectors; MIME
	7	**Application**	User Application Services	User Data	Application data	DNS; NFS; BOOTP; DHCP; SNMP; RMON; FTP; TFTP; SMTP; POP3; IMAP; NNTP; HTTP; Telnet

The best way to understand any complex system is to break it down into pieces and then analyze what they do and how they interact. Models are useful because they help us understand difficult concepts and complicated systems. When it comes to networking, there are several models that are used to explain the roles played by various technologies, and how they interact. Of these, the most popular and commonly used is the Open Systems Interconnection (OSI) Reference Model.

While “pinging” a system, layers 1, 2 and 3 are in action. Each layer defines its own “format of protocol options” known as a header. It precedes the actual data to be sent and has information such as the source IP, destination IP, checksum, type of protocol etc. And when it is attached with the actual data to be sent, it forms a protocol data unit which is renamed as per the layer.

1.6 What is ICMP used for?

The number one use of ICMP is for reporting errors. Anytime two devices are connected through the internet, ICMP can be used to create errors that can go from the receiving device to the sending device if some of the data did not arrive as expected. For example, extremely large packets of data may be too big for a router to manage. In that case, the router will discard the data packet and transmit an ICMP message to the sender informing it of the issue.

1.7 How Does ICMP Work?

To clarify the position of ICMP working principle, let us give an examplification. In TCP, the two devices that are communicating first engage in a handshake that takes several steps. After the handshake has been completed, the data can be transferred from the sender to the receiver.

ICMP is different. No connection is formed. The message is simply sent. Also, unlike with TCP and UDP, which dictate the ports to which information is sent, there is nothing in the ICMP message that directs it to a certain port on the device that will receive it.Hence, ICMP messages are divided into two broad categories which are called error messages and query messages.

1.8 ICMP Message Types

ICMP messages are divided into two general categories: error messages that are used to report problem conditions, and informational messages that are used for diagnostics, testing and other purposes.

Error Messages:

These messages are used to provide feedback to a source device about an error that has occurred. Errors are usually related to the structure or content of a datagram, or to problem situations on the internetwork encountered during datagram routing.

Informational (Query) Messages:

The IP software on a host or router has encountered a problem processing an IP datagram. For example, it may be unable to route the datagram to its destination.

Here we can see below, 8 as the Type of request which symbolizes Echo request. When one system pings another system, it sends a Type 8 request and if the host is alive, the host sends back Type 0 (Echo Reply) request.

Message Class	Type Value	Message Name	Definition	RFC
ICMPv4 Error Messages	3	Destination Unreachable	Indicates that a datagram could not be delivered to its destination. The Code value provides more infor- mation on the nature of the error.	792
	4	Source Quench	Lets a congested IP device tell a device that is sending it datagrams to slow down the rate at which it is sending them.	792
	5	Redirect	Allows a router to inform a host of a better route to use for sending datagrams.	792
	11	Time Exceeded	Sent when a datagram has been discarded prior to delivery due to expiration of its Time To Live field.	792
	12	Parameter Problem	Indicates a miscellaneous problem (specified by the Code value) in delivering a datagram.	792
ICMPv4 Informational Messages	0	Echo Reply	Sent in reply to an Echo (Request) message; used for testing connectivity.	792
	8	Echo (Request)	Sent by a device to test connectivity to another device on the internetwork. The word "Request" sometimes appears in the message name.	792
	9	Router Advertisement	Used by routers to tell hosts of their existence and capabilities.	1256
	10	Router Solicitation	Used by hosts to prompt any listening routers to send a Router Advertisement.	1256
	13	Timestamp (Request)	Sent by a device to request that another send it a timestamp value for propagation time calculation and clock synchronization. The word "Request" sometimes appear in the message name.	792
	14	Timestamp Reply	Sent in response to a Timestamp (Request) to provide time calculation and clock synchronization information.	792
	15	Information Request	Originally used to request configuration information from another device. Now obsolete.	792

1.9 ICMP and Port Dependency

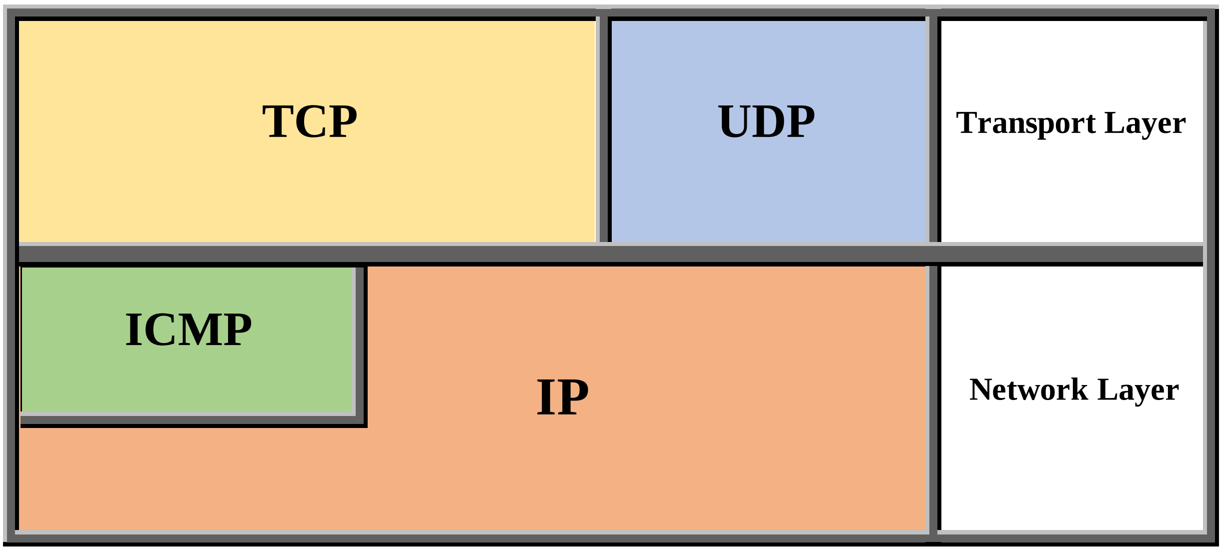
Unlike TCP and UDP, which specify the ports to which information is sent, **ICMP does not specify a specific port to which information will be received.** Ports are not the only things that communicate in TCP/IP. With TCP and UDP we have ports, because we have services listening at those ports.

If we turn off the ports, there’s still IP traffic going to the machine there is just nothing listening, or at least nothing that can hear anything.

For instance, if I’m talking to you, and you hear me, you get the information that I am telling to you, and you can respond to me, or not. But if you’re deaf, or your ears are completely plugged, you cannot hear me anymore. Does that stop you from reading something? Does it stop you from using any of your other senses for input? Of course not. Just like that example, still ICMP is cannot be blocked even if we close those ports. There must be some buildin firewalls or extentional closed ports to cut down all ICMP communicaton.

Also, ICMP is a network protocol which is located in layer 3. However it is not in layer 4 because there are no port numbers attached to ICMP.

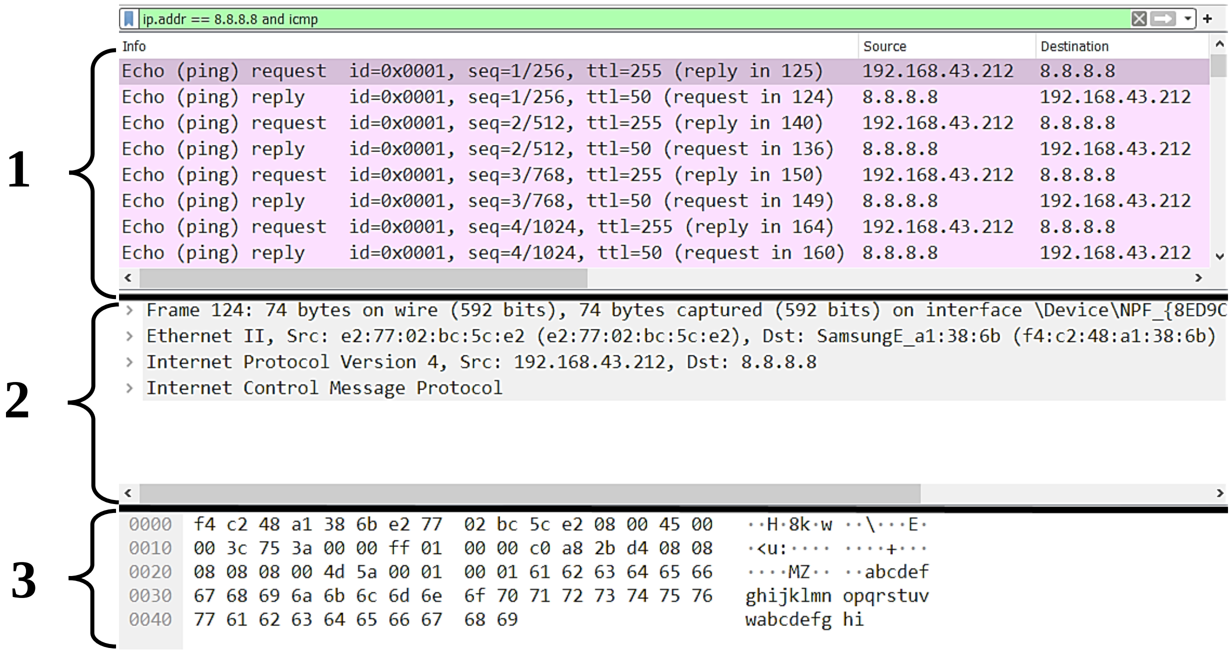
ICMP works at Network layer so has no ports and is neither **TCP** nor **UDP**.



CHAPTER 2: WIRESHARK ICMP PROTOCOL ANALYSIS

2.1 Introduction of Wireshark

Before we start the analysis, let us talk about the Wireshark program initially. Wireshark is a network packet analyzer that lets us examine the traffic flowing into and out of our Network. Wireshark is used to troubleshoot networking problems. It converts the data stream to a listing of packets flowing in and out of the computer. It allows us to examine an individual packet, and through the layers of encapsulation until the application-level payload is revealed.



```
ip.addr == 8.8.8.8 and icmp

Info                                                            Source          Destination
Echo (ping) request  id=0x0001, seq=1/256, ttl=255 (reply in 125)   192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=1/256, ttl=50 (request in 124)  8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=2/512, ttl=255 (reply in 140)   192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=2/512, ttl=50 (request in 136)  8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=3/768, ttl=255 (reply in 150)   192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=3/768, ttl=50 (request in 149)  8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=4/1024, ttl=255 (reply in 164)  192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=4/1024, ttl=50 (request in 160) 8.8.8.8         192.168.43.212
```

```
> Frame 124: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{8ED9C
> Ethernet II, Src: e2:77:02:bc:5c:e2 (e2:77:02:bc:5c:e2), Dst: SamsungE_a1:38:6b (f4:c2:48:a1:38:6b)
> Internet Protocol Version 4, Src: 192.168.43.212, Dst: 8.8.8.8
> Internet Control Message Protocol
```

```
0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Part 1:

This part is a colorful listing of all of the packets captured. Each line is a summary of a single frame or packet that was captured. The colors represent a coding scheme that can be used to quickly detect the type of packet. For example, light pink is the color for ICMP packets.

Part 2:

When we click one of the packets in area 1, the packet structure is directly has shown in area 2.

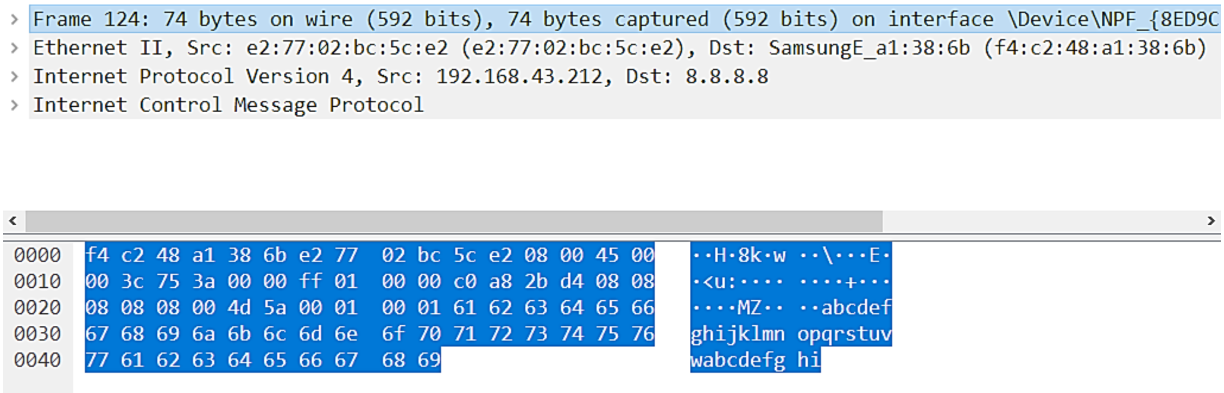
Part 3:

Clicking on a portion of the packet in area 2, changes the display in area 3. Area 3 has two parts. On the left are sixteen columns of two characters each. This is the raw hexadecimal code that makes up the packet. On the right is the Unicode version of this hexadecimal code.

2.2 Understanding Packet Details

The purpose of the section is to look at various points in the OSI layers, which can act as an aid in troubleshooting network problems. It can also help us to understand what sort of traffic is going over the network.

As we can see, the issued ICMP packet format includes Ethernet + IPv4 + ICMP informations due to the Data Encapsulation has been shown below. There are 4 parts which are Frame, Ethernet II, IPv4 and ICMP.



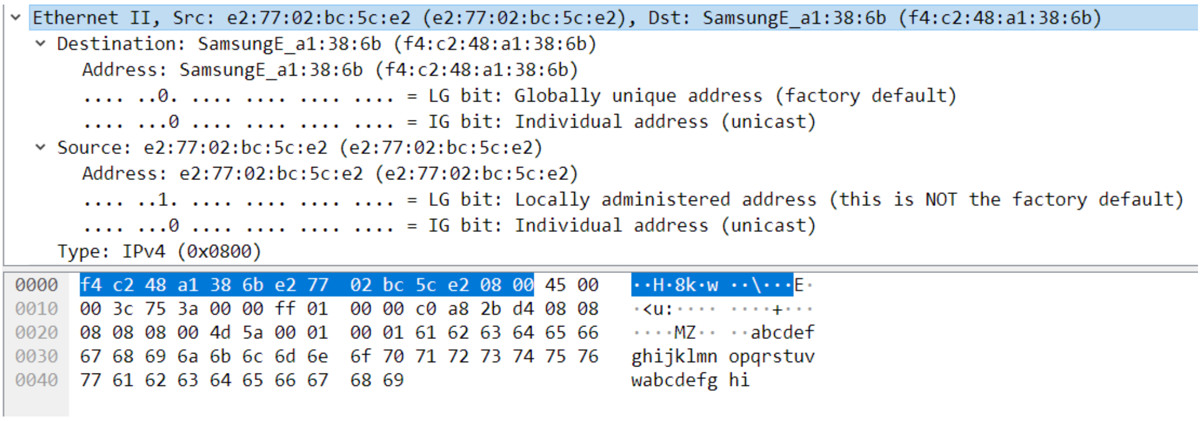
```
> Frame 124: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{8ED9C
> Ethernet II, Src: e2:77:02:bc:5c:e2 (e2:77:02:bc:5c:e2), Dst: SamsungE_a1:38:6b (f4:c2:48:a1:38:6b)
> Internet Protocol Version 4, Src: 192.168.43.212, Dst: 8.8.8.8
> Internet Control Message Protocol

0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Looking at the above figure, we can see that the first line shows a summary of the frame. The other lines show the data link layer, the network layer, the transport layer, and finally, the actual data contained within the frame. Let us have a look at the each layer in detail.

2.2.1 Data Link Layer

It contains a destination address and a source address. It is only concerned with getting a frame to the next adjacent node on the physical medium. The main point of interest is the Organizational Unique Identifier (OUI), which takes up the first three bytes of an Ethernet address. Both the LG bit (sometimes also referred to as UL bit) and the IG bit are located in the most significant byte of each MAC address.

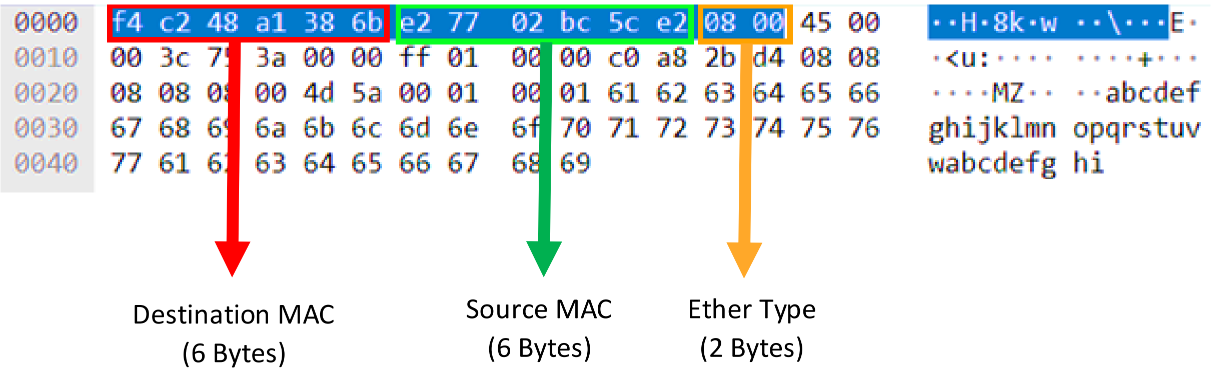


```
v Ethernet II, Src: e2:77:02:bc:5c:e2 (e2:77:02:bc:5c:e2), Dst: SamsungE_a1:38:6b (f4:c2:48:a1:38:6b)
  v Destination: SamsungE_a1:38:6b (f4:c2:48:a1:38:6b)
      Address: SamsungE_a1:38:6b (f4:c2:48:a1:38:6b)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: e2:77:02:bc:5c:e2 (e2:77:02:bc:5c:e2)
      Address: e2:77:02:bc:5c:e2 (e2:77:02:bc:5c:e2)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)

0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

As can be seen from the above figure, the 14-byte Ethernet II part has a destination address of f4:c2:48:a1:38:6b, a source address of e2:77:02:bc:5c:e2, and a data payload of 0x0800, which is IPv4. Therefore, the data frame of an ICMP ping request should be shown in the following as f4 c2 48 a1 38 6b e2 77 02 bc 5c e2 08 00.

Ethernet frame sent with ping includes the source and destination MAC along with the type of protocol being used.



```
0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

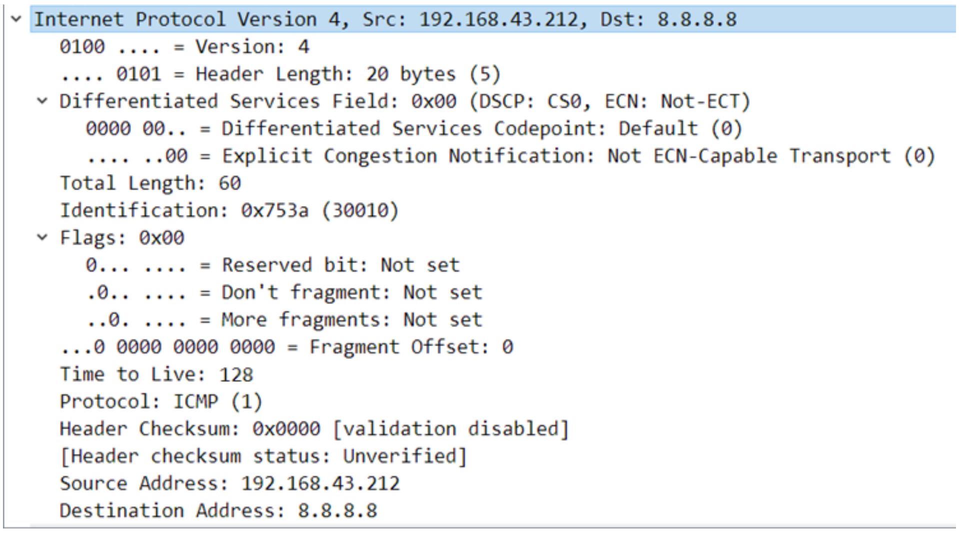
Here it is shown rest of the ether types. The most important field here is the EtherType field as on analyzing these bytes we are able to find the protocol that is used in the communication.

Ether Type	Value (in Hex. form)
ARP (Address Resolution Protocol)	0x0806
Ipv4 (Internet Protocol Version 4)	0x0800
Ipv6 (Internet Protocol Version 6)	0x086dd

2.2.2 Network Layer

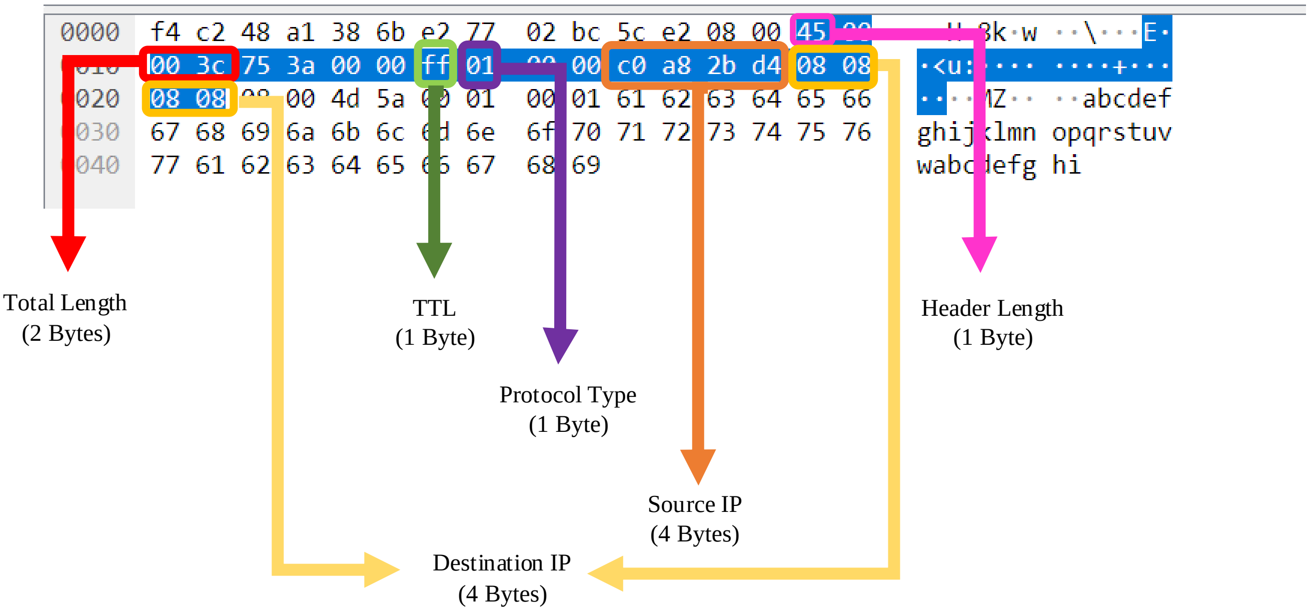
The IP layer is concerned with moving between networks. Highlighting the network layer shows more details. From figure below, we can see the source and destination IP addresses as well as the IP header length (20 bytes in this case).

We can also see the Differentiated Services (DiffServ) area. This would be where extra information relating to the packet's type of service goes. There are several other levels exist as well. Diff Serv levels can be used when looking to implement a Quality of Service (QoS) on IP networks. since ping is using IP protocol to check if a host is alive or not.



```
⌄ Internet Protocol Version 4, Src: 192.168.43.212, Dst: 8.8.8.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ⌄ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 60
    Identification: 0x753a (30010)
  ⌄ Flags: 0x00
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.43.212
    Destination Address: 8.8.8.8
```

Here is the observation of the hex dump of IP packet:



```
0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

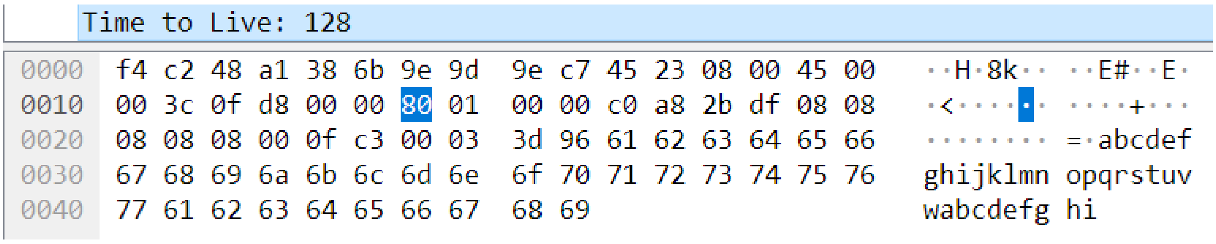
From the analysis of this IP packet, we observed that IP header is then added to the Ethernet frame to make a packet that has the above-mentioned options specified.

By analyzing this data we can observe protocol type as well:

Protocol Type Name	**Value in Hex.**
ICMP	1
TCP	6
EGP	8
UDP	11

Here below, we have adjusted the TTL value for ICMP request as 128 due to the sending the packets from Windows OS. IP Time-to-Live Field Value with ICMP. TTL field value with ICMP has two separate values: one for ICMP query message and one for ICMP query replies.

The TTL field value helps us identify certain operating systens abd groups of operating systems. It also provides us with the simplest means to add another check criteria when we are querying other host(s) or listening to traffic (sniffing).



```
Time to Live: 128

0000  f4 c2 48 a1 38 6b 9e 9d  9e c7 45 23 08 00 45 00   ··H·8k·· ··E#··E·
0010  00 3c 0f d8 00 00 80 01  00 00 c0 a8 2b df 08 08   ·<······ ····+···
0020  08 08 08 00 0f c3 00 03  3d 96 61 62 63 64 65 66   ········ =·abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

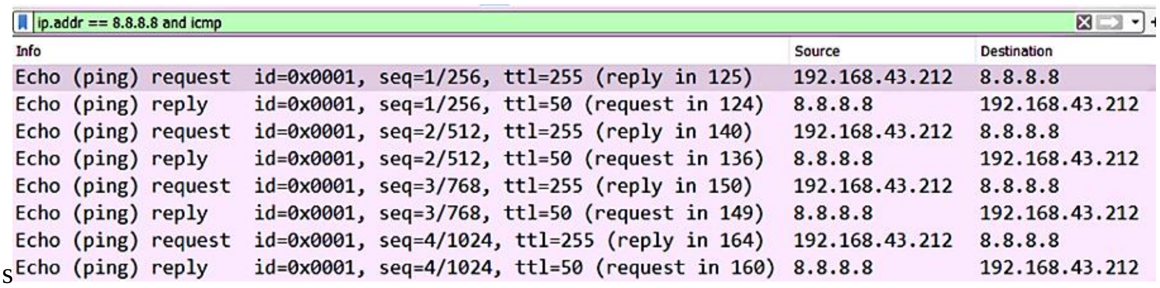
2.3 ICMP Analysis Details

A datagram is completed and transported when ICMP header is added in Layer 3. IP lacks error control mechanism, so ping uses ICMP to serve that purpose. ICMP protocol is specified in IP header as we saw above and hence, it is important to analyse ICMP header to better understand the underlying mechanism of ping.

When a datagram is sent from source to destination, in our case, when we ping, the format of the datagram is as follows:

Layer 2	Layer 3	
Ethernet Header Attached	IP Header Attached	ICMP header and payload data attached

We have configured the filtering entry in Wireshark initially which is "ip.addr == 8.8.8.8 and icmp". Then we started using it to grab packets. After that, 4 consecutive sample ICMP request packets were sent. Each set of two messages is containing the ICMP response request packet of the gateway sent by the local host and the reply packet sent by the gateway to the local host. We can see the captured packets in Wireshark, as shown in the following figure.

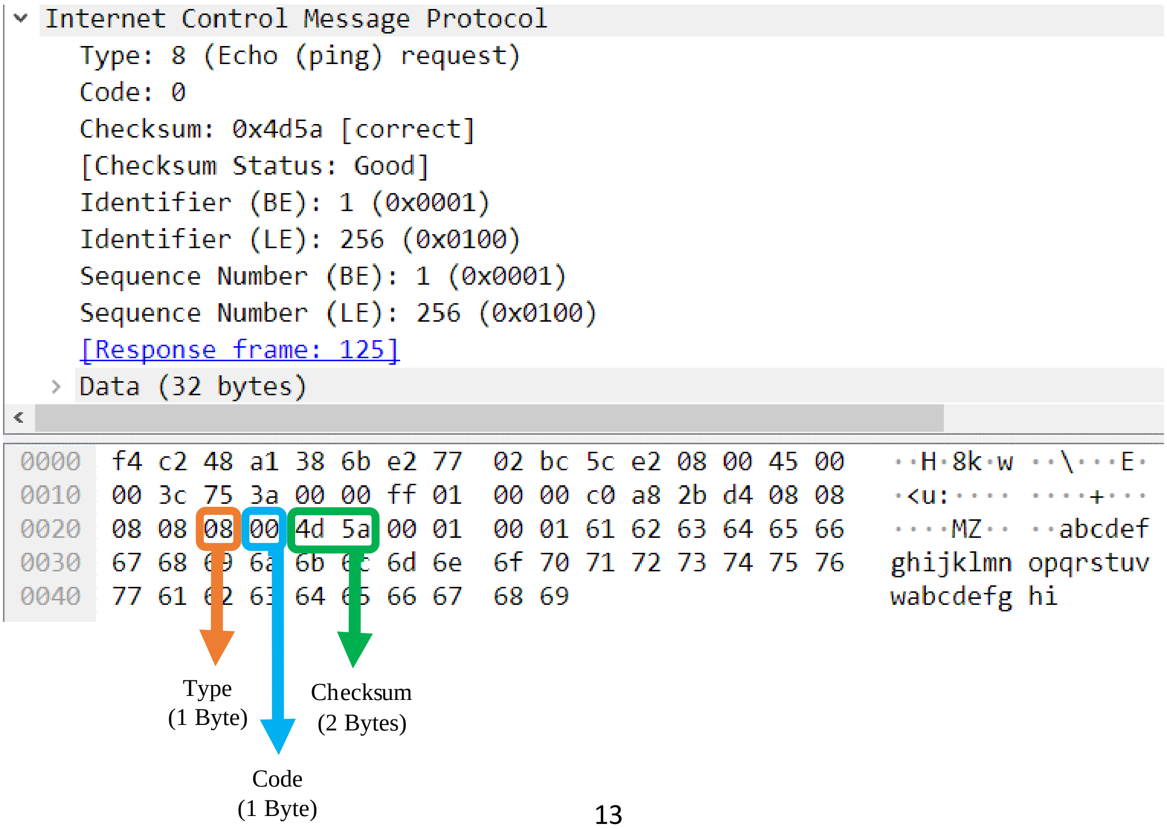


ip.addr == 8.8.8.8 and icmp

```
Info                                                                        Source          Destination
Echo (ping) request  id=0x0001, seq=1/256, ttl=255 (reply in 125)        192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=1/256, ttl=50 (request in 124)       8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=2/512, ttl=255 (reply in 140)        192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=2/512, ttl=50 (request in 136)       8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=3/768, ttl=255 (reply in 150)        192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=3/768, ttl=50 (request in 149)       8.8.8.8         192.168.43.212
Echo (ping) request  id=0x0001, seq=4/1024, ttl=255 (reply in 164)       192.168.43.212  8.8.8.8
Echo (ping) reply    id=0x0001, seq=4/1024, ttl=50 (request in 160)      8.8.8.8         192.168.43.212
```

s

Here is the observation of the hex dump of ICMP packet:



```
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d5a [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 125]
  > Data (32 bytes)

0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

The **Type** Field indicates the type of ICMP message. An Echo request message will have the number 8 in the Type field and an Echo Reply message will have the number 0.

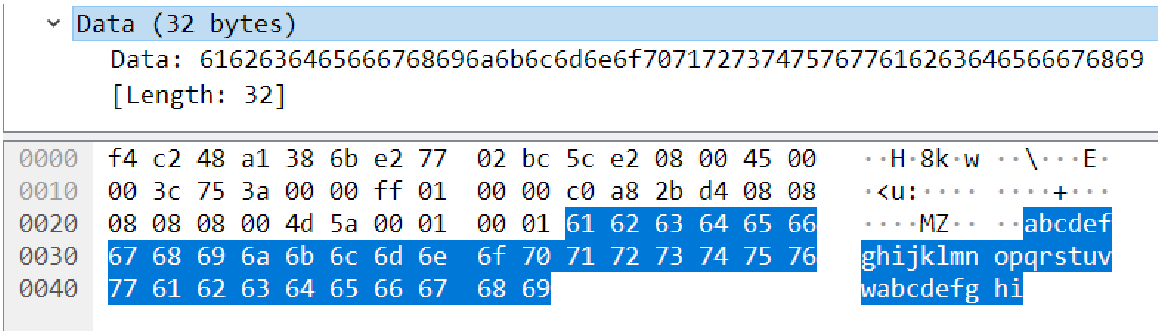
The **Code** Field is used by the different message formats to indicate specific error conditions. For Echo, the code field is always 0.

The **Checksum** is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type.

Note that an ICMP packet **does not have source and destination port numbers** because it was designed to communicate network-layer information between hosts and routers, not between application layer processes.

In RFC792, the document outlining the ICMP protocol that ICMP, uses the basic support of IP as if it were a higher level protocol. However, ICMP is actually an integral part of IP, and must be implemented by every IP module. This is why since every IP module needs it**, ICMP is a network protocol which is located in layer 3.** However it is not in layer 4 because there are no port numbers attached to ICMP.

Here below is the hexadecimal content of the 74 Bytes data packet has been shown. Usually, it contains a default Payload Data such as this ASCII string — "abcdefghijklmnopqrstuvwabcdefghi".



```
  ˅ Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f7071727374757677616263646566676869
      [Length: 32]

0000  f4 c2 48 a1 38 6b e2 77  02 bc 5c e2 08 00 45 00   ··H·8k·w ··\···E·
0010  00 3c 75 3a 00 00 ff 01  00 00 c0 a8 2b d4 08 08   ·<u:···· ····+···
0020  08 08 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

The data portion is mandatory in reply packets, if there is a data portion in the request packet. The answering system must send a reply containing exactly the same data portion. The RFC792 does not specify if the data portion in the request packet is mandatory. The Data in this schema is not marked as (optional) as in many other RFCs. Note that, RFC 792 says that 8 bytes of payload should be included.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ⎫
|     Type      |     Code      |          Checksum             |  ⎪
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ⎪
|           Identifier          |        Sequence Number        |  ⎬   ICMP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ⎪   Header
|     Data . . .                                                   ⎪
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ⎭
```

The data is the same in the echo reply and request. The echo request sender can use any convenient data, and the echo reply sender will copy its data from the request so that the payload returns to the original sender.

However, sending a ping with zero bytes of data is entirely possible as well.

When a ping command is sent to the host, the datagram is encapsulating Ethernet header, IP header, ICMP header and payload too. The minimum size of an IPv4 header is 20 bytes and the maximum size is 60 bytes.

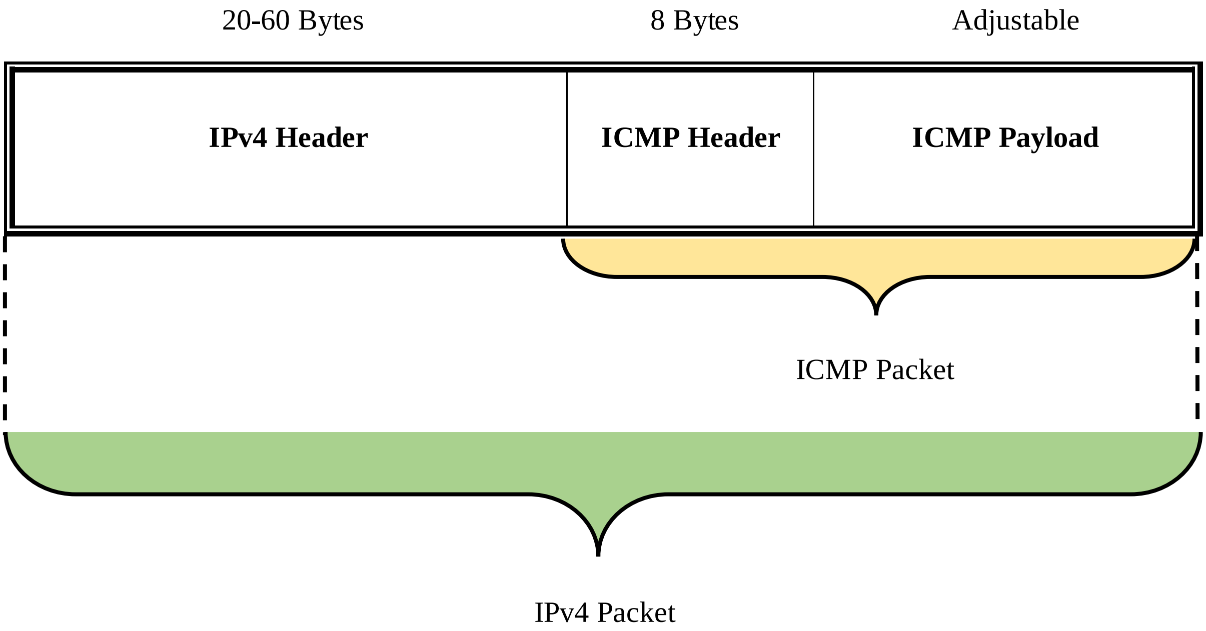
The default size of payload data in ping in windows is 32 bytes. We have added 20 bytes of IP header in it and 8 bytes of ICMP header as well. 32+20+8, it comes out to be 60 bytes.

On the other hand, when we analyse ping in Wireshark, the size of the frame written in the log is 74 bytes. This is because while pinging, we would need the destination and source MAC address as well, which is available in the Ethernet header.

To put in a nutshell, in total:

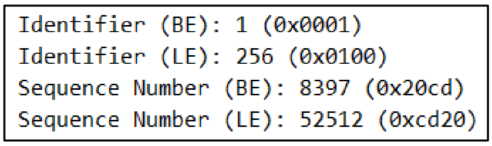
Ethernet header	IP header	ICMP header	ICMP payload size
14 Bytes	20 Bytes	8 Bytes	32 Bytes (default)

ICMP encapsulation summary:



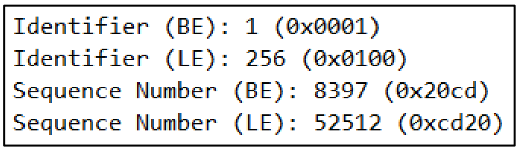
Next comes an Identifier and Sequence Number field. These fields are used to link echo request and reply packets together. Let us have a look at the Identifier and Sequence Number values for the request and response below. As we can see both of the verified as same:

Request Message:



```
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence Number (BE): 8397 (0x20cd)
Sequence Number (LE): 52512 (0xcd20)
```

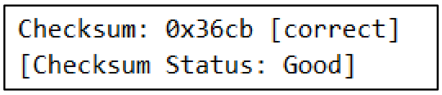
Response Message:



```
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence Number (BE): 8397 (0x20cd)
Sequence Number (LE): 52512 (0xcd20)
```

The Identifier and Sequence Number may be used by the echo sender to aid in matching the replies with the echo requests. For example, the identifier might be used like a port in TCP or UDP to identify a session, and the sequence number might be incremented on each echo request sent.They returns these same values in the echo reply with the same value.

The Type and Code is followed by a 16 bit checksum over the compelete ICMP message, to verify that it was received correctly. Its column has shown below:



```
Checksum: 0x36cb [correct]
[Checksum Status: Good]
```

Note that Wireshark may show these fields in two ways: as a Big Endian (BE) value and a Little Endian (LE). The difference is the order in which the bits are organized into bytes, e.g., 00000001 on the wire might represent “1” or “256” depending on whether the first bit transmitted is the least (LE) or most (BE) significant bit.

Each echo request and corresponding echo reply have the same Identifier value and the same Sequence Number value. The values are used to match the echo request to the right echo reply. Typically, the Identifier is kept the same and the Sequence Number is incremented. This ensures that as a pair, echo requests will have different Identifier/Sequence Number values so they (and their corresponding replies) can be distinguished and verified.

CHAPTER 3: METHOD ANALYSIS

In order to carried out the analysis, there were two types of sockets used to get more sophistication and practise.

The first method was to use a predefined protocol in the IP header, such as ICMP.dll method.

The second method was to use a custom protocol in the IP header. Raw sockets provide the ability to manage down-level transports.

3.1 ICMP.dll Method

The IcmpSendEcho function is exported from the Icmp.dll on Windows 2000. The IcmpSendEcho function is exported from the Iphlpapi.dll on Windows XP and later.

In the old days, pinging a computer on windows required building an ICMP packet from scratch and using Raw Sockets to send the packet to its destination. We then had to listen for a response and parse this manually.

Microsoft since Windows 2000 made available an ICMP helper library so it is not a requirement anymore to initiate with a raw socket method. On the other hand, the handicap is that this method is dependent on Windows OS. It mainly benefits from Win32API to handle process.

The IcmpSendEcho function send an ICMP echo request to the specified address and returns the number of replies received and stored in ReplyBuffer. IcmpSendEcho function is exported from the Iphlpapi.dll which provides all the functions of the ICMP protocol. Those are,

- IcmpHandle, (HANDLE)
- DestinationAddress, (unsigned long)
- RequestData, (LPVOID)
- RequestSize, (DWORD)
- ReplyBuffer, (LPVOID)
- ReplySize, (DWORD)
- Timeout, (DWORD)

The easiest way to implement PING programmatically on the Windows platform is to call iphlpapi.dll dynamic link library, which references the following functions:

IcmpCreateFile	Opens a handle on which IPv4 ICMP echo requests can be issued.
IcmpSendEcho	Sends an IPv4 ICMP echo request and returns any echo response replies and the call returns when the time-out has expired or the reply buffer is filled.
IcmpCloseHandle	Closes a handle opened by a call to the IcmpCreateFile or IcmpCreateFile functions.

The implementation steps are as follows:

1. Use the function IcmpCreateFile to create an Icmp handle.
2. Construct the API parameters. (IcmpHandle, ipaddr, SendData, sizeof(SendData), &ipOptions, ReplyBuffer, ReplySize, Timeout)
3. Call the function IcmpSendEcho to send.
4. Close the Icmp handle obtained by IcmpCreateFile.

3.1.1 Parameter Details

IcmpHandle	The open handle returned by the IcmpCreateFile function.
DestinationAddress	The IPv4 destination address of the echo request, in the form of an IPAddr structure.
RequestData	A pointer to a buffer that contains data to send in the request.
RequestSize	The size, in bytes, of the request data buffer pointed to by the RequestData parameter.
ReplyBuffer	A buffer to hold any replies to the echo request. Upon return, the buffer contains an array of ICMP_ECHO_REPLY structures followed by the options and data for the replies. The buffer should be large enough to hold at least one ICMP_ECHO_REPLY structure plus RequestSize bytes of data.
ReplySize	The allocated size, in bytes, of the reply buffer. The buffer should be large enough to hold at least one ICMP_ECHO_REPLY structure plus RequestSize bytes of data. This buffer should also be large enough to also hold 8 more bytes of data (the size of an ICMP error message).
Timeout	The time, in milliseconds, to wait for replies.

3.2 Raw Socket Method

The basic concept of low level sockets is to send a single packet at one time, with all the protocol headers filled in by the program (instead of the kernel).

A raw socket is a type of socket that allows access to the underlying transport provider. To use raw sockets, an application needs to have detailed information on the underlying protocol being used. The socket type has been used, however, was SOCK_RAW, which includes the IP headers and all protocol headers and data as well.

There is also a separate module for the IP checksum calculation function while using for ICMP because it is not specifically embedded to pinging; this same algorithm is used in other parts of TCP/IP.

A raw socket is a type of socket that allows access to the underlying transmission protocol. It can directly send data from the application layer to the network layer, and parse the protocol at the network layer.

Standard socket() call used to create a raw socket includes:

- Family is AF_INET, as for TCP or UDP
- Socket type is SOCK_RAW instead of SOCK_STREAM or SOCK_DGRAM
- Socket protocol needs to be specified, e.g. IPPROTO_ICMP (often left at 0 for UDP or TCP sockets)

  Thus, the function we use as, Socket (AF_INET, SOCK_RAW, IPPROTO_ICMP)

The implementation steps are as follows:

1- Create a socket of type SOCK_RAW and set the protocol IPPROTO_ICMP, and set the properties of the socket.

2- Create and initialize ICMP header.

3- Call the sendto function to send an ICMP request to the remote host.

4- Call the recvfrom function to receive any ICMP response**.**

3.3 Summary Algorithm For Both Methods

To conclude the Ping Algorithm for both methods as:

1- Initialize echo request

2- Send echo request

3- Wait for echo reply (or time out)

4- Receive reply

5- Report results

6- Go back to loop to execute for sending Icmp request again until complete.

CHAPTER 4: BIBLIOGRAPHY

- https://www.rfc-editor.org/rfc/rfc792
- https://www.rfc-editor.org/rfc/rfc1122
- https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#_live_capture_from_many_different_network_media
- The TCP/IP Guide - Charles M. Kozierok
- https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml
- http://www.csc.villanova.edu/~mdamian/Past/networksfa12/Notes/ICMP.pdf
- https://networkdirection.net/articles/network-theory/icmptypes/
- https://web.ecs.syr.edu/~wedu/Teaching/cis758/LectureNotes/ICMP.pdf
- https://www.dazhuanlan.com/shaofa/topics/1692502
- https://blog.securityevaluators.com/icmp-the-good-the-bad-and-the-ugly-130413e56030
- https://networkdirection.net/articles/network-theory/icmptypes/
- https://notes.shichao.io/tcpv1/ch8/#chapter-8-icmpv4-and-icmpv6-internet-control-message-protocol
- https://resources.infosecinstitute.com/topic/icmp-protocol-with-wireshark/
- https://www.jianshu.com/p/4bd8758f9fbd
- http://mail.imach.uran.ru:3000/p/SOURCE/WSSAMPLE/WEBPAGE/winsock2.htm
- https://commons.wikimedia.org/wiki/File:ICMPv4_encapsulation-en.svg
- https://acikders.ankara.edu.tr/pluginfile.php/63511/mod_resource/content/1/OSI%20Katmanlar%C4%B1.pdf
- https://docs.microsoft.com/en-us/windows/win32/api/icmpapi/nf-icmpapi-icmpsendecho
- https://chromium.googlesource.com/external/libjingle/chrome-sandbox/+/60598307c80be80da28e5ae7921352bd874fb05b/talk/base/winping.cc
- http://mail.imach.uran.ru:3000/p/SOURCE/WSSAMPLE/WEBPAGE/ms_icmp.htm
- https://courses.cs.vt.edu/cs4254/fall04/slides/raw_1.pdf