## Practice 2

Wireshark IP and ICMP protocols analyses

## Background:

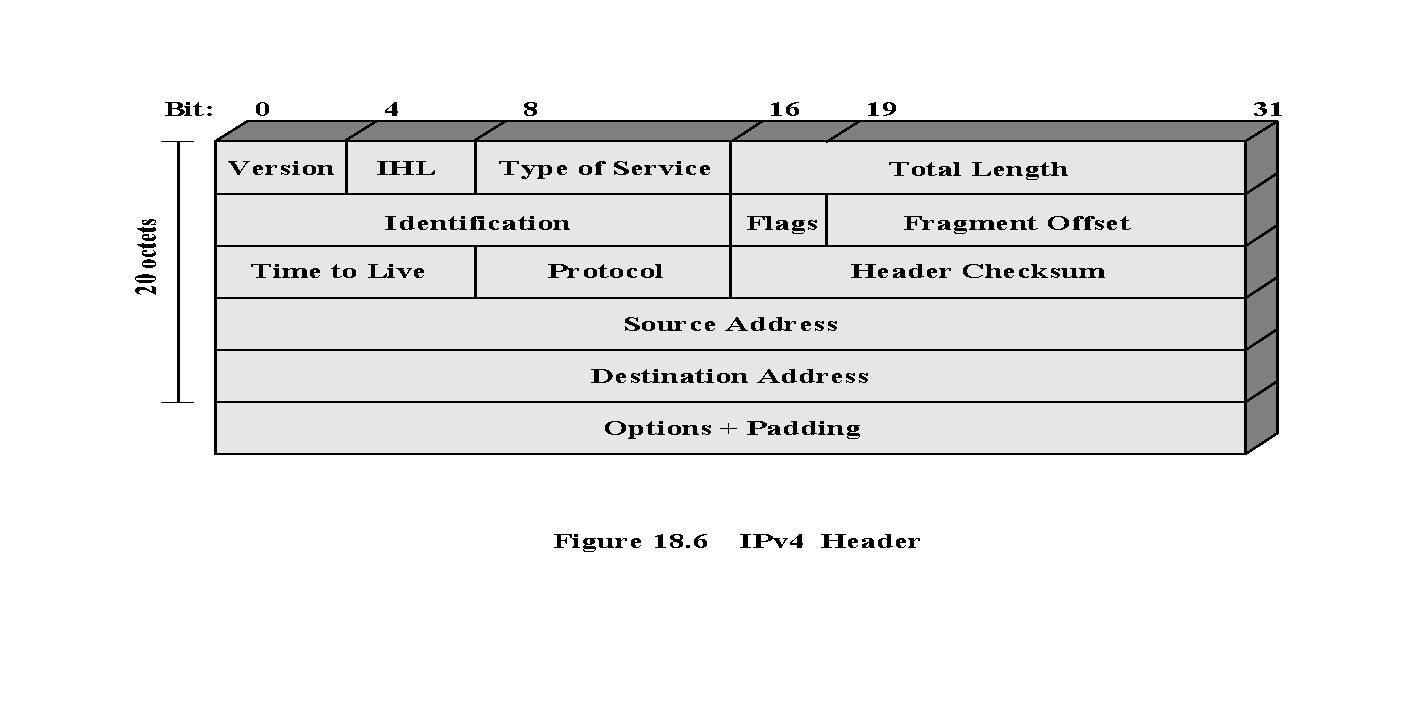
* The IP header and its fields
* IP fragmentation
* ICMP Protocol

## Introduction

The most important objective of the IP protocol in networks that follows the TCP/IP architecture is to send information from an origin host to a destination host without an assured direct connection. For this, the IP protocol includes a useful header with extra data making it possible for the intended data to reach its destination. For accomplishing it, IP makes use of other protocols like ARP, ICMP or IGMP. Particularly, ICMP sends query or error messages related to different hosts IP.

**Fundamentals: IP and ICMP headers. Ping and Tracert commands.**

The IP header structure is shown in Figure 1.



*Figure 1: IP header structure*

Structure summary:

| **Field** | **Example Value** | **Description** |
| --- | --- | --- |
| *Version* | 4 | IP protocol version (4 for us). |
| *IHL* | 5 | **I**nternet **H**eader **L**ength as 4 bytes words. Without *Options*, its regular value is 5 (20/4). |
| *Type of Service* | 0000 | Transmission rate, priority, delay, throughput…parameters. Normally is not used -- its value is 0. |
| *Total Length* | 1500 | Packet length. Its maximum value is 65535 bytes. Ethernet networks have a value of 1500, as it is the MTU value. |
| *Identification* | 0x8302 | Unique packet identifier. It is used in fragmentation. |
| *Flags* | 000 | 3 bits, of which just the last two, DF (**D**o not **F**ragment) and MF (**M**ore **F**ragments), are used. |
| *Fragment Offset* | 185 | It tells the exact position where to locate the fragment’s data when reassembling the datagram (using 8 bytes words). |
| *Time to Live* | 64 | It tells the network how many routers (hops) this packet can cross. |
| *Protocol* | 6 | It tells to which protocol this packet belongs to. Each protocol is associated to a number, some of which are TCP = 6, UDP = 17, ICMP = 1, IGMP = 2, and IPv4 = 4. They all can be found at: <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml> |
| *Checksum* | 0xd206 | It keeps the checksum value of the entire header which is then used to check if the packet is received error-free (Wireshark does it automatically). |
| *Source Address* | 192.168.1.1 | 32-bit address of the sender (or source) of the packet |
| *Destination Address* | 150.214.18.1 | 32-bit address of the Receiver (or destination) of the packet. |
| *Options* |  | Variable size (0-40 Bytes) optional field. It is usually empty. |

**NOTE:** In many cases what Wireshark shows in the details of the header, is not exactly what is being transmitted. This is due to Wireshark pre-processing the data. For example, observe the header length and fragment offset fields.

Some of the ICMP messages are shown in Figure 2.



*Figure 2: Possible ICMP header structures (incomplete)*

All possible messages include the following fields:

| **Field** | **Example Value** | **Description** |
| --- | --- | --- |
| Type[[1]](#footnote-0) | 11 | Message type (Echo Reply, Echo Request, Time Exceeded, etc.). |
| Code1 | 0 | Details the purpose of the message. |
| Checksum | 0xfb74 | Checksum for error-free validations. |

The additional information of the ICMP message depends on the message type. For example, when it is a “*time exceeded*” the respective IP header is included alongside the first 8 bytes of the readable packet load.

The **ping** command is used to check if a host is running or not. It makes use of the ICMP protocol allowing to: edit some IP header fields, and set the packet size. The first **ping** argument is the name (or IP address) of the target host. The most important parameters options are as follows:

| **Win** | **Linux** | **Mac** | **Description** |
| --- | --- | --- | --- |
| **-l** size | **-s** size | **-s** size | ICMP message data field size. |
| **-i** ttl | **-t** ttl | **-m** ttl | IP’s TTL (*time to live*) field value. |
| **-f** | **-M do** | **-D** | Enables DF bit flag (don’t fragment) |
| **-n** numb | **-c** numb | **-c** numb | How many ICMPs are sent. For our exercises use **–n 1** or **–c 1** (i.e., one) |
| **-r** num | **-R** | - | Stores in the header options the IPs of the hosts through which the ICMP message passes (both send and return). |

The **tracert** command (traceroute in Linux/Mac) gets the list of intermediate nodes between the source and the destination hosts. Some nodes dismiss tracert, hence, it will return an incomplete list of intermediate nodes. **tracert** argument is the identifier of the destination host (URL or IP address). Again, in Mac OS **tracert** is always executed with -I option.

**Task 1: IP encapsulation.**

**Step 1: Wireshark configuration for frames capturing.**

Configure Wireshark before network capturing. Open it as an administrator, and choose the proper network interface. To begin packet capture, select in the menu **Capture -> Options**, and then click the “Start” button (and Details button if you want additional information about the interface) of the network card. Note: **disable the promiscuous mode**, to only capturing either incoming or outgoing traffic to/from your computer (it should be disabled by default, see it by opening the Options button of the Interfaces window).

**Step 2: Generate network traffic to and capture the session in the file p2e1-2.pcapng.**

Note: If captured packages are not as expected, try again after flushing ARP/DNS (Windows example):

*Own computer version* (admin allowed):

arp –d

netsh interface ip delete arpcache

ipconfig /flushdns

Laboratory version (admin locked):

ipconfig /flushdns

**First** you will open a web-browser private window (e.g., incognito mode in Firefox) and access our faculty website ([**http://www.lcc.uma.es/**](http://www.lcc.uma.es/)). **Second**, open a terminal of Windows (i.e. “Simbolo del sistema”), or click **Inicio > Ejecutar**, write **cmd**, enter its context menu and right-clicking open it as an administrator. In the terminal window **ping** to [**www.informatica.uma.es/**](http://www.informatica.uma.es/) with options -n 1 (Windows) or -c 1 (rest)**. Third**, and in the same command prompt, run ipconfig /renew. When the load finishes, stop capturing and save the frame as **p2e1-2.pcapng**.

**Step 3: Analyse the data captured by Wireshark.**

Filter the capture to only show the frames related to the icmp, dns, dhcp and http protocols. Then, observe any of the filtered frames of each protocol identifying every IP header fields (see Figure 1). After analysing the frames captured with Wireshark, answer the following question:

**Exercise 1.** Observe the different packets datagram IP headers. Which is the protocol in the field “protocol” of the datagrams headers that transport ICMP, DHCP, DNS and HTTP messages? Fill the following table with the proper information.

What is the meaning of this field? The protocol to which the information is passed.

Why does this field have the same value if the application protocol is different?

| Protocol | Protocol Field Value  (text) | Protocol Field Value  (HEX) | #Frame |
| --- | --- | --- | --- |
| ICMP | ICMP | 01 | 5723 |
| HTTP | TCP | 06 | 715 |
| DNS | UDP | 11 | 624 |
| DHCP | UDP | 11 | 4928 |

**Exercise 2.** Select the ICMP request (echo request message) and fill in the table with the IP header destination address and the MAC destination address (i.e., the Ethernet header). Do the same for an DNS request (the info field shows “*Standard query 0x…*”).

|  | ICMP | DNS |
| --- | --- | --- |
| IP destination address (IP header) | 150.214.57.91 | 150.214.40.11 |
| MAC destination address (Ethernet header) | c4:b3:6a:0a:2e:75 | c4:b3:6a:0a:2e:75 |
| Frame Number | 5723 | 624 |

*Why are the MAC destination addresses the same as opposed to the IP destination addresses?* Because MAC destination addresses are of the same node: the router. IP addresses are from different nodes.

**Task 2: IP Fragmentation**

**Step 1: Wireshark configuration for frames capturing.**

Configure Wireshark before network capturing. Open it as an administrator, and choose the proper network interface. To begin packet capture, select in the menu **Capture -> Options**, and then click the “Start” button (and Details button if you want additional information about the interface) of the network card. Note: **disable the promiscuous mode**, to only capturing either incoming or outgoing traffic to/from your computer (it should be disabled by default, see it by opening the Options button of the Interfaces window). Also, please check if the following option is disabled (otherwise, Wireshark will not show the fragments):

*Edit > Preferences > Protocols - IPv4 > Reassemble fragmented IPv4 datagrams*

**Step 2: Generate network traffic by ping to www.informatica.uma.es with different options and capture the session in the file p2e3.pcapng.**

The ping command allows editing some IP packets fields. Specifically, we can edit the sending ICM message size. This helps us to test how the IP fragmentation works. Open a terminal of Windows (i.e. “Simbolo del sistema”), or click **Inicio > Ejecutar**, write **cmd**, enter its context menu and right-clicking open it as an administrator. In the terminal window **ping** to [**www.informatica.uma.es**](http://www.informatica.uma.es) with 2 values for the message size: first ping with size 1300 bytes and then a second ping with size 3400 bytes. Additionally, use options -n 1 (Windows) or -c 1 (Linux/Mac)**.** When **ping** finishes, stop capturing and save the frame as **p2e3.pcapng**.

**Step 3: Analyse the data captured by Wireshark.**

**Exercise 3.** What are the type and code of the ICMP messages (request/reply)? Focus on ICMP requests for the rest of the questions. What filter can you use to only show the fragments of a concrete IP datagram? Fill in the following table with the value of the active flags of each fragment, their identification and their offset. Note: for each message size, in case of several fragments, write one value per fragment separated by commas (,)

| Size | #Frame | Identifiers (Big Endian) | Flags | Fragment Offsets |
| --- | --- | --- | --- | --- |
| 1300 | 120 | 0x0001 | 0x0 | 0 |
| 3400 | 498 | 0x0001 | 0x1 | 0 |

IP datagrams have a maximum size depending on the Maximum Transmission Unit (MTU) of the data link level of the output link. For instance, the MTU of Ethernet 2 is 1500 Bytes, but the datagram contains less user data due to some bytes being reserved for the headers. To validate the maximum data size that we can send in an IP datagram, we use the DF (**D**on’t **F**ragment) bit flag. Please note that If you set this flag with a large size than the maximum possible, the packet must fail (failure ICMP message). The **ping** -f option (-d in Mac and -m in Linux) allows activating this flag. You can check the MTU of the network interfaces of your computer using the command **netsh interface ipv4 show subinterfaces** in Windows (**ifconfig** in Mac and **ifconfig** or **ip –c address** in Linux).

**Exercise 4.** Calculate the maximum amount of data that you can transmit in our laboratory network. Being MAX the maximum amount of user data that you have calculated, **ping** two times to **www.informatica.uma.es** with sizes MAX and MAX+1 (-l option), and always with the DF bit flag active (-f option). What is the value of MAX? Why? Does the first ping appear in the Wireshark packet frame? And in the second ping? Why? Save the traffic capture as **p2e4.pcapng**.

*MAX = 1472.*

*The first ping appears in the Wireshark packet frame, but not the second one, because the last one has failed.*

**Task 5: tracert**

**Step 1: Wireshark configuration for frames capturing.**

Configure Wireshark before network capturing. Open it as an administrator, and choose the proper network interface. To begin packet capture, select in the menu **Capture -> Options**, and then click the “Start” button (and Details button if you want additional information about the interface) of the network card. Note: **disable the promiscuous mode**, to only capturing either incoming or outgoing traffic to/from your computer (it should be disabled by default, see it by opening the Options button of the Interfaces window).

**Step 2: Ping to www.informatica.uma.es using different options.**

As mentioned above, the ping command allows you to modify certain fields of the IP packet. Now try pinging **www.informatica.uma.es** once and use the options -r 1, -r 3, and -r 9. This ping option -r X, makes use of the IPv4 "Record Route"[[2]](#footnote-1) service that stores the IPs it passes through, where X is the number of IP addresses that can be stored. This task cannot be performed on a Mac and is also slightly different on Linux (uses the -R option without parameters). Save the trace as **p2e5-6.pcapng**.

**Exercise 5**. Using -r X changes the header in two ways: it adds the X-dependent options field of appropriate size and therefore changes the HLEN field. How does the size of HLEN increase according to X? If you try other X values, you will see that it only allows values between 1 and 9, why do you think it only allows those values and not larger? Finally, notice that in addition to the IP option "Record Route", the option "End of Options List" is added to indicate that there are no more options, why is it necessary to add this option and not just the HLEN?

*For -r 1 -> HLEN = 28 bytes*

*For -r 3 -> HLEN = 36 bytes*

*For -r 9 -> HLEN = 60 bytes*

**Exercise 6**. Locate and observe a response packet and pay attention to the TTL field. How much is it worth? Compare it with the TTL of the request message. Who sets each value?

*Frame 127: TTL = 63 (reply)*

*Frame 126: TTL = 128 (request)*

**Task 4: Error-reporting messages of the ICMP protocol**

**Step 1: Wireshark configuration for frames capturing.**

Configure Wireshark before network capturing. Open it as an administrator, and choose the proper network interface. To begin packet capture, select in the menu **Capture -> Options**, and then click the “Start” button (and Details button if you want additional information about the interface) of the network card. Note: **disable the promiscuous mode**, to only capturing either incoming or outgoing traffic to/from your computer (it should be disabled by default, see it by opening the Options button of the Interfaces window).

**Step 2: Ping to www. informatica.uma.es with different TTL and capture the session in the file p2e7.pcapng.**

The easiest method to provoke the generation of an error is by making use of the ICMP *time exceeded*. We can do this by assigning a very low value to the IP header TTL field. The ping command allows doing so. When **ping** finishes, stop capturing and save the frame as **p2e5.pcapng**. Now open a terminal of Windows (i.e. “**Simbolo del sistema**”), or click **Inicio > Ejecutar**, write **cmd**, enter its context menu and right-clicking open it as an administrator. In the terminal window **ping** to [**www.informatica.uma.es**](http://www.informatica.uma.es) with options **-n 1** (Windows) or **-c 1** (Linux/Mac), and additionally **–i** in Windows or **-c** in Linux/Mac with different values for the TTL field.

**Exercise 7. ping** several times to **www.informatica.uma.es** incrementing the TTL, starting with a value of 1, and stopping when receiving successful answers from the server. Test the following values (stop when the answers are correct): 1, 2, 3, …. Observe in Wireshark the packets exchange. Which ICMP message is received when the packets do not arrive (type, code and meaning of that message)? What does that ICMP message contain as additional data (within the data field)?

*Type 11 (0x0B); code 0 (Time to live exceeded in transit).*

**Task 5: tracert**

**Step 1: Wireshark configuration for frames capturing.**

Configure Wireshark before network capturing. Open it as an administrator, and choose the proper network interface. To begin packet capture, select in the menu **Capture -> Options**, and then click the “Start” button (and Details button if you want additional information about the interface) of the network card. Note: **disable the promiscuous mode**, to only capturing either incoming or outgoing traffic to/from your computer (it should be disabled by default, see it by opening the Options button of the Interfaces window).

**Step 2: Use tracert to obtain information from the intermediate routers.**

The **tracert** command (**traceroute** in Linux/Mac) allows discovering how many hops are needed to reach a host, and also obtains information of the intermediate routers. After cleaning the ARP and DNS cache, use **tracert** to find out how many hops are since your computer to **www.informatica.uma.es** while capturing the frame with Wireshark. Remember to use the option -I in Mac for ICMP messages. Save the capture as **p2e8.pcapng.**

**Exercise 8.** Which type of packets (i.e., highest level protocol) does **tracert** make use for discovering the hops? Besides the path messages, tracert can provoke other auxiliary sendings in order to obtain information or prettify it, which other messages can be necessary? What strategy does **tracert** use to find out which router (i.e., IP address) belongs to each packet hop?

1. The types/codes list can be found at <https://www.rfc-es.org/rfc/rfc0792-es.txt> [↑](#footnote-ref-0)
2. <https://www.geeksforgeeks.org/options-field-in-ipv4-header/> [↑](#footnote-ref-1)