v.1.2

Universidad de León
Ingeniería Informática
Course on Computer Networks

Homework On Essential Protocol Stacks and traceroute

Details about this homework submission are published in the agora

ICMP protocol messages can be used for purposes other than for checking IP connectivity as we did in the introductory sections of the practice about Network Interfaces. ICMP offers a message type that helps to estimate the network pathway from a source *end system*[1] to a destination *end system*. Employing ICMP for this purpose involves the use of a field of IP packets named TTL or *Time To Live* which was conceived for the purpose of discarding any IP packet ever after it loops repetitively around a number of routers. These routing loops habitually are due to configuration errors on some IP routers located amid the path that currently is being visited by the packets sent by the source host to the destination host. TTL helps alleviating the problem of packets entering a routing loop.

RFC 791 specifies the behavior expected from an IP router when it receives a new IP packet as it relates to the IP packet's TTL field:

"Since every module that processes a datagram must decrease the TTL by ***at least one*** *even if it processes the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist."*

In summary, RFC 791 prescribes that whenever an IP router receives an IP packet, it must decrement its TTL field by *at least 1* (TTL = TTL - 1) so it will eventually reach 0, that is, if the number of routers visited by the packet suffices. Furthermore, the IP specification prescribes that whenever the TTL reaches 0, the router should discard the packet altogether. This action of discarding a packet is conventionally known as *dropping the packet.*

TTL was conceived as an upper bound to the number of routers an IP packet should cross on its trip to its destination, thus, in the case a router configuration error caused a network loop, packets would deterministically be dropped, *soon*. A looping packet wastes a variety of network and compute resources, and consequently it has to be avoided at all costs.

The TTL field can be applied to uses beyond the dropping of looping IP packets. Indeed, clever use of the TTL field of IP packets can help us *estimate* the sequence of routers that make up the *current* path from a source Internet host to some destination host. The traceroute utility aims at that by sending a *probe IP packet,* an IP packet that contains a null-payload UDP datagram from UDP port 33434 (By default) having a TTL value 1. This probe IP packet, upon being received by the first IP router on the pathway that leads to the destination host chosen by the user, will compute a TTL 0; the router must immediately send back an ICMP *Time Exceeded Message (TEM)* to the source host (The host

[1] Oftentimes, end system is used instead of the familiar word host or Internet host.

running the traceroute utility). The sent TEM, when received by the source host becomes a kind of notification about the presence of that router on the path from source to destination. The source host has *revealed* a new router on the network path that leads to the destination. At this point, the challenge for the source host consists of profitably interpreting the contents of the TEM to the goal of building the path.

Can the TEM received by the source host stand out for the *revealed* router? Conceptually, the identity of the revealed router is the IP address of the router interface used for transmitting the TEM. That IP address is contained in the source IP address field of the IP packet that encapsulates the TEM. Thus, determining the revealed router identity is straightforward.

The TEM message sent by the revealed router is an ICMP message having code 0 and type 11 that carries a payload composed of the following two parts:

- 1. The *header* of the IP probe packet which was received by the revealed router
- 2. The *header* of the UDP datagram carried by the IP probe packet as its own payload

`traceroute` utilities from Linux or Unix systems, typically continue by sending a probe IP packet containing TTL 2, which will elicit a TEM from the second router on the pathway. This process of sending probe packets having monotonically growing values of TTL continues until it is the destination host itself the revealed *router*. `traceroute` will finish at this point.

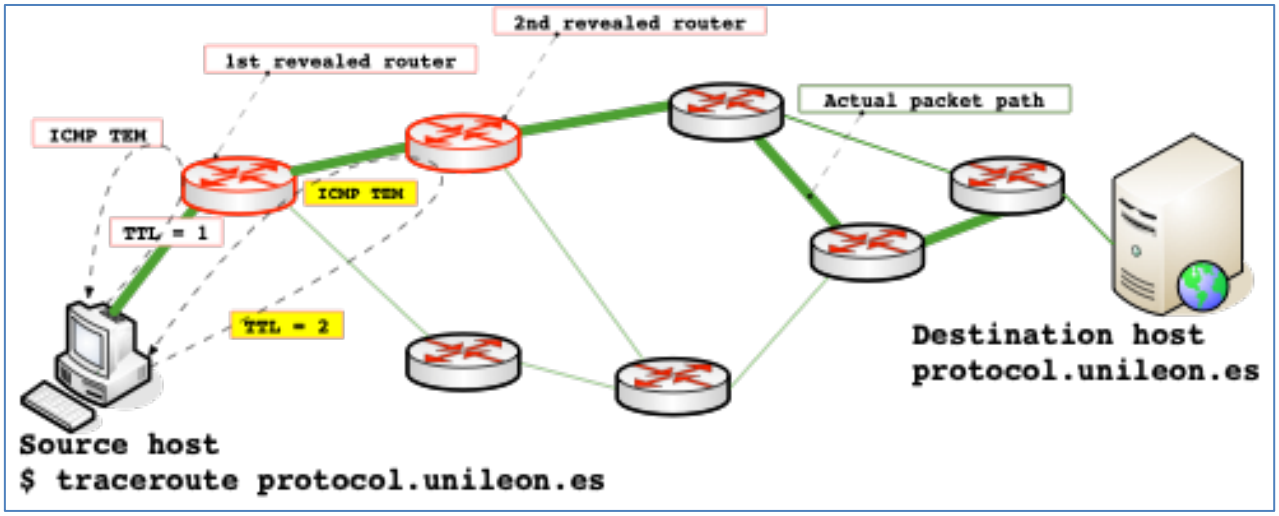

Figure 1. Source host sending two probe packets having TTL=1 and TTL=2 and receiving the respective TEM messages

Exercise 1.

Included in the provided *.zip* file is file *trace.pcapng* which contains a trace that will serve us for analyzing the IP packets sent and received by the `traceroute` utility. Run Wireshark and load that trace file by using the File|Open menu. When the trace has been loaded, Wireshark will display all of the frames contained in it. Since we are interested in observing only the frames relevant to traceroute, we'll limit the frame display by applying

the following *display filter:*

```
(udp.port>=33000 && udp.port<34000) || icmp.code==0 && !ssdp && !dns
```

The filter is applied by copying the preceding text and then pasting it on the text box on the upper area of the Wireshark Graphical User Interface (GUI) and by clicking on the right-arrow button contained therein. Applying this filter will limit the displayed frames which will greatly improve our ability to analyze the sent and the received traffic.

- a. Identify the Frame number column and tell whether gaps appear on the sequence of frame numbers

  After applying the above filter gaps do appear on the frame numbers

- b. Give an explanation for the missing frames, if any

  If the filter is not applied, then the gaps disappear and we can observe the full traffic received and sent by the host capturing the Wireshark trace. Gaps appear because the display filter restricts the traffic displayed to the frames complying with it thereby making the others disappear. Since the disappearing frames are intermixed with those complying with the filter, their sequence numbers present gaps. The following segment from the trace illustrates this point:

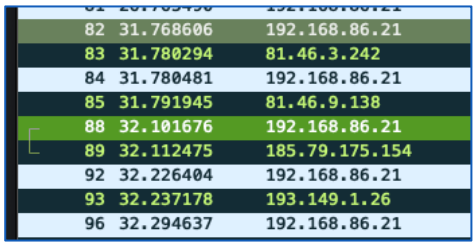

Fig 1.1. The gap from frame no. 85 to frame no. 88 indicates that some frames were sent or received which are not complying with our display filter.

Exercise 2.

The provided trace was captured at a source host housed in a Macintosh computer running OS-X; the destination host runs a Debian Linux which DNS name is protocol.unileon.es. The output obtained from the execution of traceroute appears on Fig. 2.

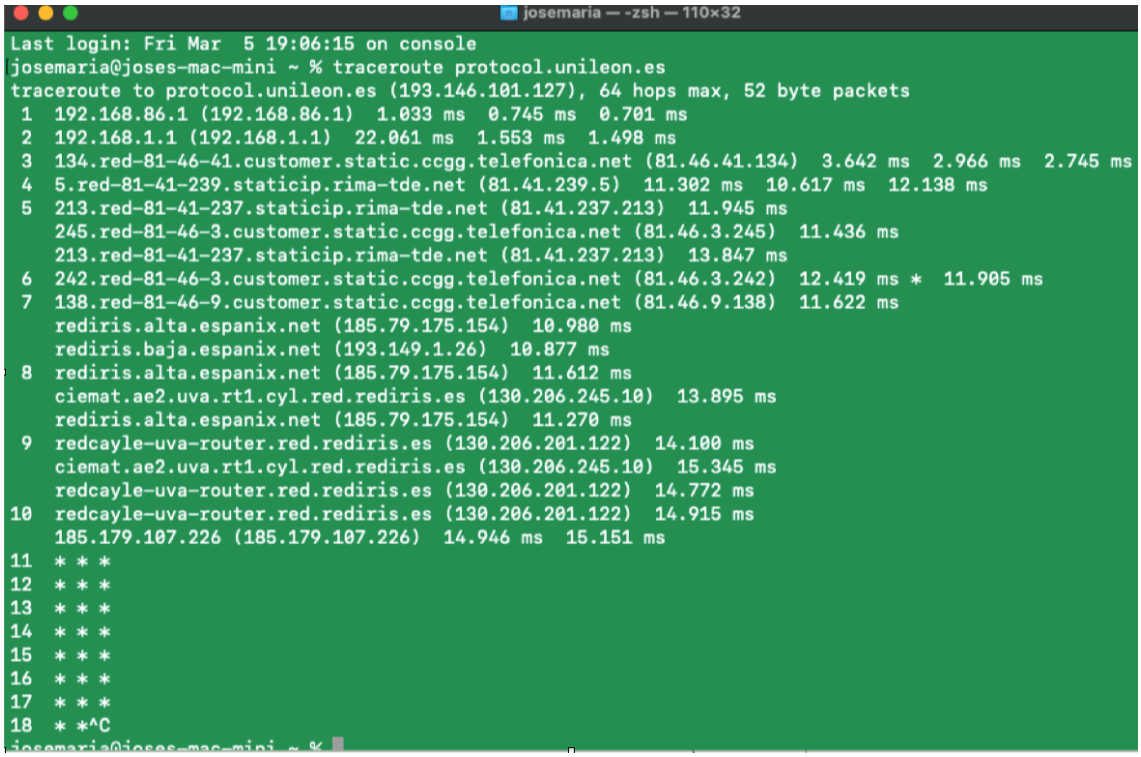

Figure 2. Output from traceroute at source host

a. According to the resulting screen output from traceroute, what's the identity of the 3rd revealed router, *that is,* the IP address of that router sending interface?

The third router's sending interface ip address is 81.46.41.134

b. What's the average Rtt (Round Trip Time) achieved by the packets sent to this router?

The three Rtts (Round Trip Times) reported by traceroute are: 3,642 ms; 2,966 ms and 2,745 ms; the average Rtt is `(3,642 + 2,966 + 2,745)/3 ms = 3,117 ms`

Exercise 3.

On Fig. 3 frame no. 55 is displayed by single clicking on it. Display it and respond to the following questions:

a. Does frame no. 55 carry an IP packet? Explain why.

Yes; frame no. 55 carries an IP packet as indicated by Wireshark identifying the encapsulated layer-2 protocol as Internet Protocol version 4 (IPv4).

b. What is the *protocol data unit* (PDU) encapsulated by the IP packet? Explain your answer.

The IP packet carried in frame no. 55 encapsulates a UDP datagram. Wireshark identifies the payload encapsulated by the IP Packet as "User Datagram Protocol". The PDU of this protocol is conventionally known as UDP datagram. Alternately, we can identify the payload

encapsulated by the IP packet by observing the IP packet's Protocol field. The contents of the Protocol field is 17, which corresponds to the UDP protocol (Search Internet for "protocol numbers" and you will obtain a search result that points to the site where official Internet numbers are kept: `https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml`)

c. What's the full protocol stack resulting from frame no. 55? The requested protocol stack is represented in Fig. 4. Obtaining it consists of reading the contents of the frame displayed by Wireshark *in reverse order* vs. the order of the layers in the Internet architecture. Examining Fig. 3 will help you understand how to obtain the protocol stack from the frame contents.

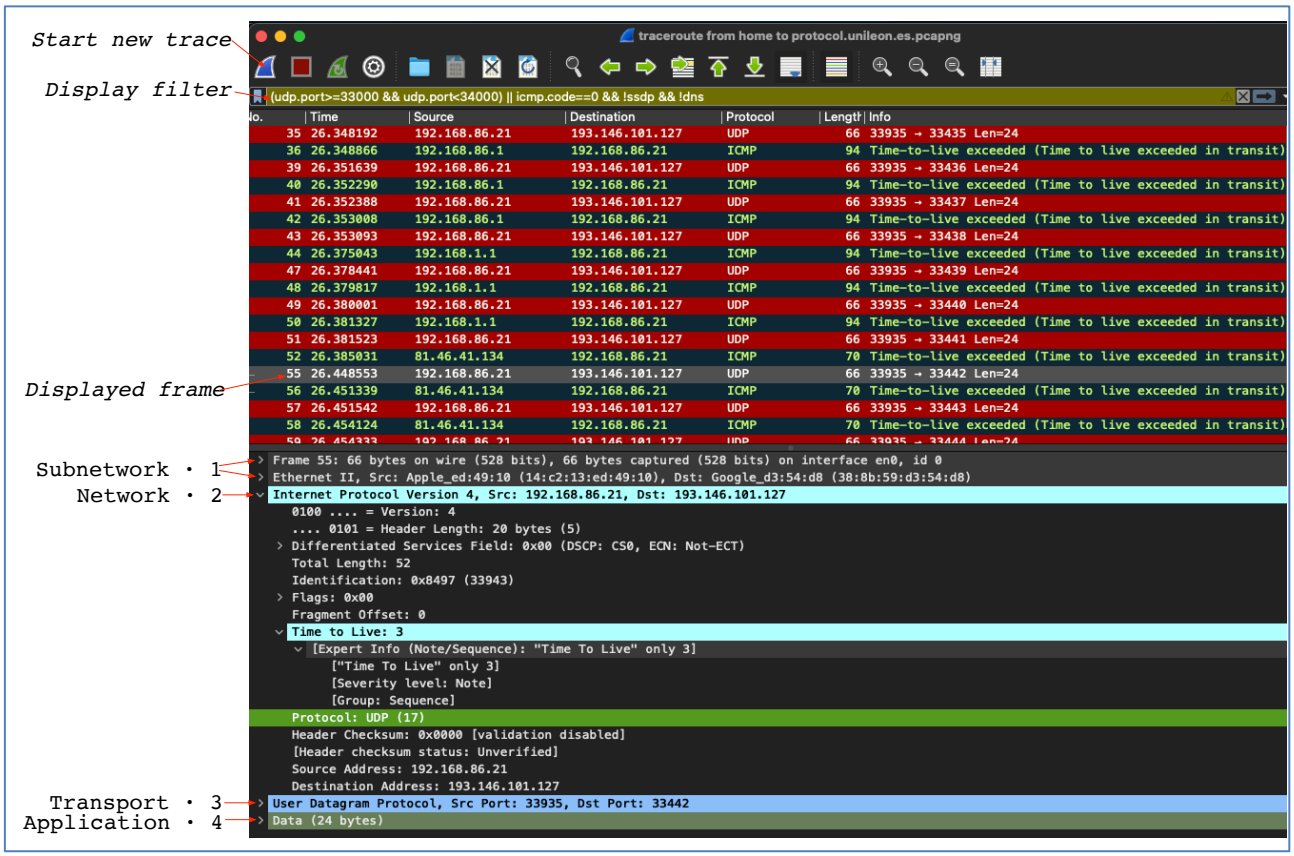

Figure 3. Output from traceroute at source host

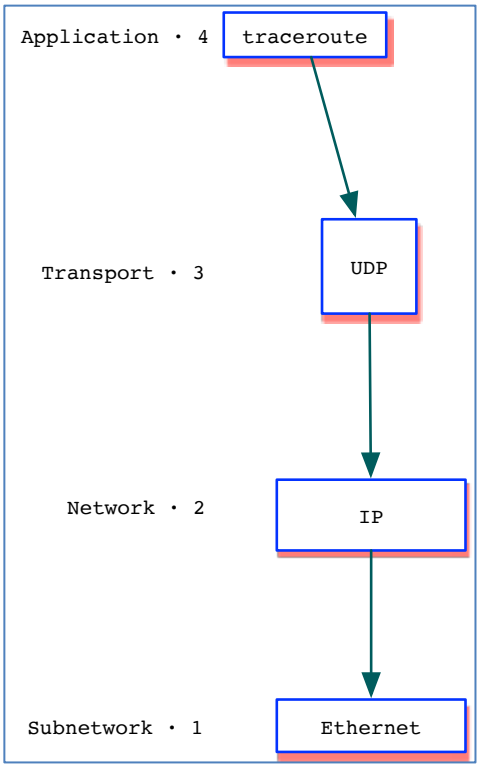

Figure 4. Protocol stack of a traceroute probe packet

Exercise 4.
Display frame no. 56 and respond to the following questions:

a. The frame carries an IP packet. Identify its Destination IP address, its Source IP address and its multiplexing key (Its name is "Protocol")

The multiplexing key of an Ethernet frame is its "Type" or "Ethertype" field, which in the case of frame no. 56 is 0x0800, consequently, the frame must carry an IP packet. You can obtain the listing of Ethertype values and their correspondences by searching "ethertype" in Internet and clicking on this search result: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml. The protocol encapsulated in an Ethernet frame when its Ethertype contains 0x0800 is IPv4.

Identifying the contents of the Destination IP Address of an IP packet is straightforward in Wireshark, you must simply unfold the IP packet and then go to its Destination IP field which in this case is 192.168.86.21; the Source Address is 81.46.41.134 and the IP Packet's Protocol field contains 1 which identifies the ICMP protocol.

b. Which protocol is encapsulated into the IP packet that is being carried by the frame? Explain your answer.

In the response to the preceding question we established the IP Packet's multiplexing key, or Protocol field as 1. The protocol encapsulated into the IP packet is ICMP.

c. Obtain the full protocol stack expressed by the displayed frame; depict it fully so that

it precisely expresses all the protocols that appear on the frame display in Fig. 5

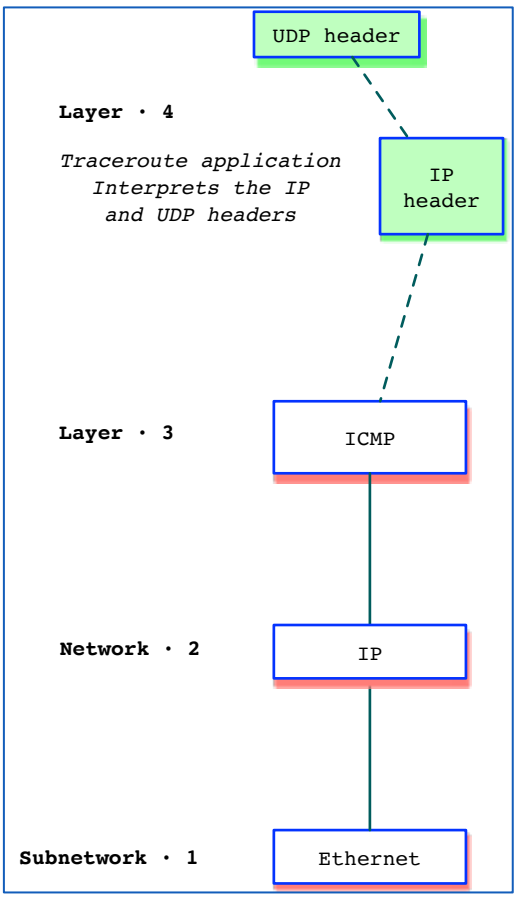

Fig. 4.1. Full protocol stack resulting from the TEM message in frame no. 56 (Fig. 5)

d. What do you think that is the justification for such a protocol stack containing *so many* encapsulated protocols.

The TEM (TTL Exceeded Message) message sent back by a router that computes a TTL of 0 must carry the IP and the UDP headers from the IP packet received by the router which TTL resulted in 0. Therefore, the TEM message encapsulates an ICMP message (TEM) along with the original IP and UDP headers which will be processed and interpreted by the traceroute application.

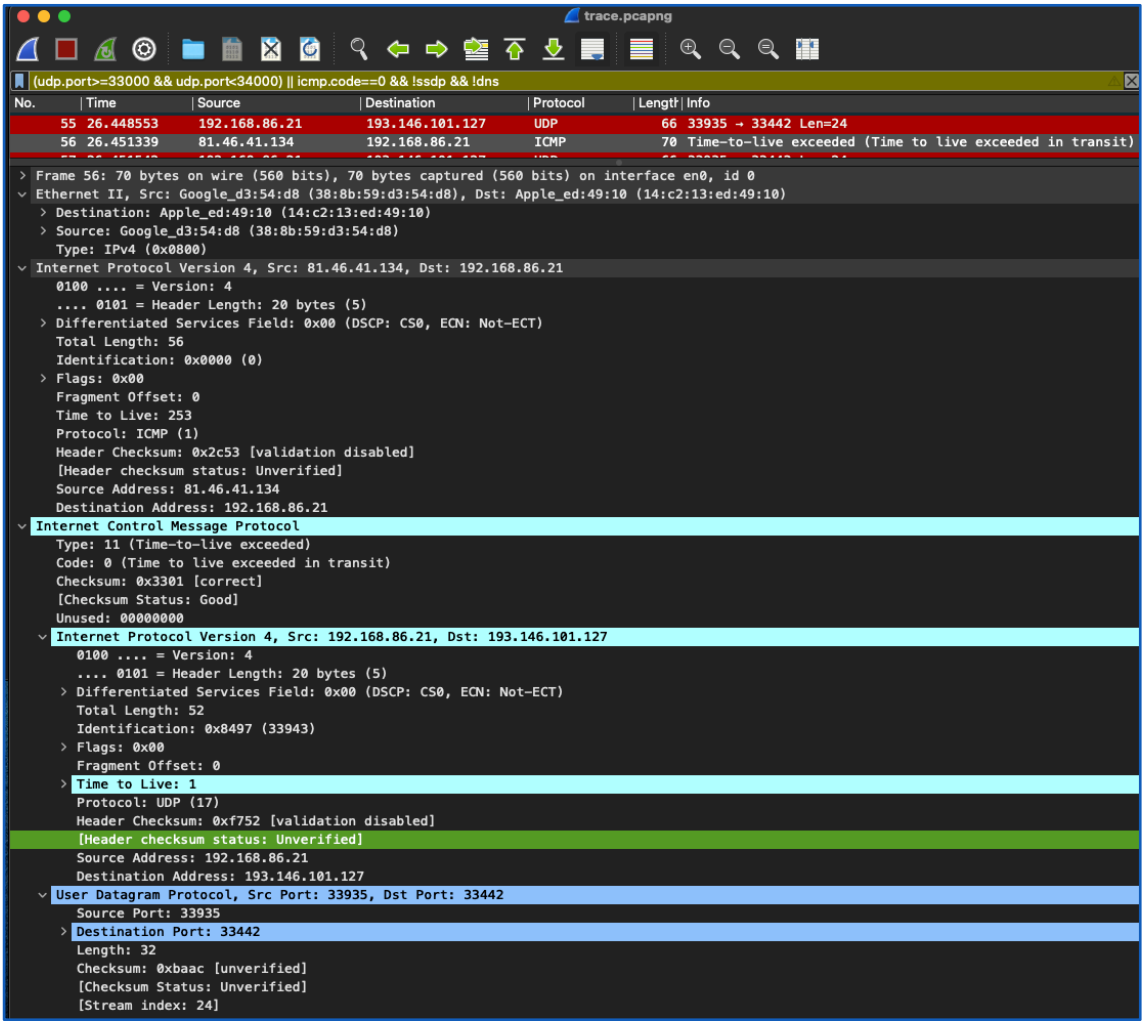

Figure 5. Display of frame no. 56 which carries a TEM (ICMP TTL Exceeded Message)