IMT Atlantique
Département Informatique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
URL: **www.imt-atlantique.fr**

UE PRIP
2023

Lab Ethernet and ARP

Version: 1.3

Report filled-in by:

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1. Objective

- Understand the purpose and functioning of the ARP protocol
- Explore differences of single-segment and multiple-segment networks

2. Pre-Lab

Beforme coming to the lab:

- make sure you have read the introduction of this lab.
- study the linux commands `arp`, `ip-neighbour`, `ifconfig`, `ping`, `route` you can refer to their man pages by typing on a console `man <command>`

3. Introduction

3.1. The Address Resolution Protocol (ARP)

When a host A tries to send data to a host B, A's network layer builds a packet with the following IP addresses: IP source= A's IP address, IP destination = B's IP address, and with the payload. That packet is afterwards passed to A's data link layer. This layer does not understand IP addresses, but it uses an address in accordance to the technology being used. In the case of Ethernet (or IEEE 802) networks, each network interface has a 48-bit address which is usually assigned by the manufacturer in a unique basis. Ethernet interfaces communicate between them using these 48-bit addresses. It is thus necessary to have some sort of mapping between IP and Ethernet addresses, and making it possible with the lowest possible overhead and administration effort. In the internet, this is achieved thanks to the Address Resolution Protocol (ARP).

ARP is defined in RFC 826 [1], it allows to solve the mapping between network layer addresses (IP addresses in TCP/IP networks) and data link layer addresses. In broadcast networks (such like Ethernet networks) the MAC (medium access control) sublayer is in charge of managing the access to the medium (either by managing turns to stations or by managing the collisions). Usually the addresses used in broadcast mediums are called MAC layer addresses or simply MAC addresses. So the 48-bit addresses of Ethernet networks are the MAC addresses of such network, and we shall use both names interchangeably.

ARP does not relay in any administration or human intervention. When A tries to communicate with B, the protocol finds out B's MAC address having as an input B's IP address.

Please note that this protocol works for IPv4. During this lab, for teaching purposes, we are going to focus on IPv4. However, in IPv6 the same problematic must be solved as well. In the case of IPv6, is the Neighbor Discovery Protocol [2], which among other functions, performs the functions equivalent to those of ARP. The `arp` command we shall use in this lab, is only available for IPv4. For IPv4 and IPv6 you can use `ip neighbour`, which we shall as well use in this lab.

3.1.1. ARP packet format

ARP messages go directly as the payload of the MAC sublayer frame (Ethernet frame for the case of Ethernet). They contain, among other fields the origin and destination IP and MAC addresses of the frame. You can see the complete format of ARP packets in Fig. 1

3.1.2. Protocol functioning

We are going to see the ARP functioning throughout the lab. In order to prepare your lab you are encourage to read about protocol functioning, for instance, in [3, 1] or browsing the Internet.

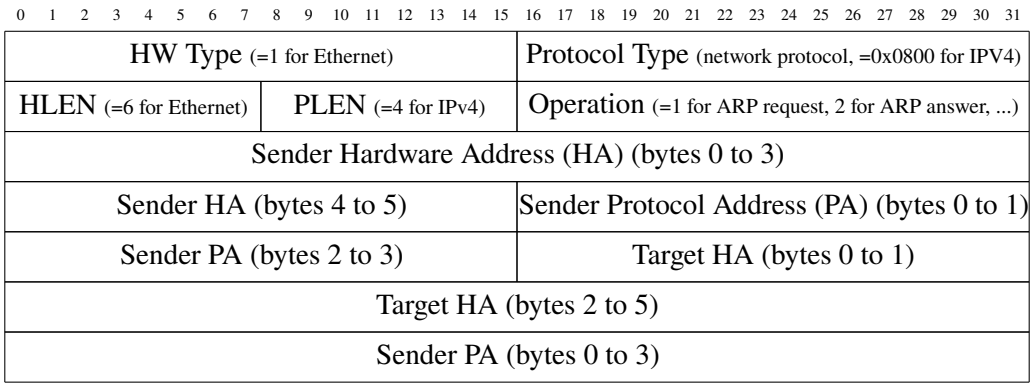

Figure 1: ARP packet format.

3.1.3. ARP cache

Machines (host, routers) stock the mapping between IP address and MAC address during a certain time, so that the protocol does not have to be executed every time a packet needs to be sent. This improves efficiency of the communication. The mapping is stored in the so-called ARP table or ARP cache. The entries in the memory are frequently erased (typically every 20 minutes) and can also be manually erased.

In Unix systems one command that allows to see the content of the ARP table is `arp -a`. See the man page for the complete documentation, you will find in particular that the command allows to delete entries of the table as well (-d ¡ip address¿) and avoid the resolution of names associated to IP addresses (-n option).

Another available command is `ip-neighbour`, for instance to see all the content of the arp cache you can type on a terminal `ip neigh show` or to delete or entries in the cash you can type `ip neigh flush all`. You can see the complete documentation by running `man ip-neighbour`.

3.2. Recall: Network addresses and Netmasks

A network interface, to work properly, needs to get configured the address itself as well as the network mask. For instance, 192.168.122.2/24 where at the left of the / we have the address and at the right the so-called netmask or subnet mask. The netmask allows to determine which part of the address corresponds to the ¡prefix¿, i.e. the common part of every address belonging to that same network.

Historically, there were five different classes of IPv4 addresses, A to E, as shown in Table **??**. Each of those classes implied a ¡prefix¿ of a different length. The prefix part, as said before, indicates the common part of every address belonging to the same network, while everything that comes after the prefix indicates the number of the machine in that network. Please note that according to the class of the address we can have a different number of machines in the network. Notice as well that this classful addressing scheme only allows for prefixes to be of one, two or three bytes, according to the class.

Later on, more flexibility was needed, i.e. allowing a customisable maximum number of machines at each network, or also sub-networking (talking some bits from the machine part of the address to identify a subnetwork). The so-called Classless InterDomain Routing (CIDR) was introduced. Netmasks are a key concept in CIDR. A netmask is a special binary number that along an IP address allows to easily determine the ¡prefix¿ or what is the same, all the addresses belonging to the same network, as we will see shortly.

Netmasks have the same bit-length as the associated IP address. The left-most bits of the netmask are set to one, in such a way that a bit-wise AND between the binary representation of the IP address and the netmask, results in the network address. In other words, the number of bits set to one in the netmask indicates the number of bits of the IP address that corresponds to the ¡prefix¿. A pair IP address/netmask determines a block of IP addresses belonging to the same network. The first address of that block is reserved to designate the network, and the last one for broadcast (i.e. for all machines in the network). The

Class	Leading bits	size of ¡prefix¿ in bits	Start address	End address	Default subnet mask in dot-decimal notation	netmask in CIDR notation
Class A	0	8	0.0.0.0	127.255.255.255	255.0.0.0	/8
Class B	10	16	128.0.0.0	191.255.255.255	255.255.0.0	/16
Class C	110	24	192.0.0.0	223.255.255.255	255.255.255.0	/24
Class D (multicast)	1110	not defined	224.0.0.0	239.255.255.255	not defined	not defined
Class E (reserved)	1111	not defined	240.0.0.0	255.255.255.255	not defined	not defined

Table 1: IPv4 classes

IP address	Netmask CIDR notation	Netmask dot-decimal notation	Network address	Broadcast address
192.168.122.109	/24	255.255.255.0	192.168.122.0	192.168.122.255
192.168.122.109	/30	255.255.255.252	192.168.122.108	192.168.122.111
10.4.4.4	/16	255.255.0.0	10.4.0.0	10.4.255.255

Table 2: Examples of netmasks

remaining addresses are available for machines. Some examples are shown in Table 2.

The concept of netmask exists and is used both in IPv4 and IPv6.

4. Hands on

Throughout this lab we are going to work with the network shown in Fig. 2.

Question 4.1.

Explain with words the different components of the network in Fig 2 (imagine you are describing the network topology to someone that is not looking at the picture). Make sure you understand your own words!

4.1. The Address Resolution Protocol

4.1.1. Set-up the network topology

1) Run the course VM

2) Check the presence of the `net_labs` directory, if it is present, update your copy of the repository:

```
cd net_labs
git pull origin master
```

If it is not present, clone the repository and move to the `net_labs` directory:

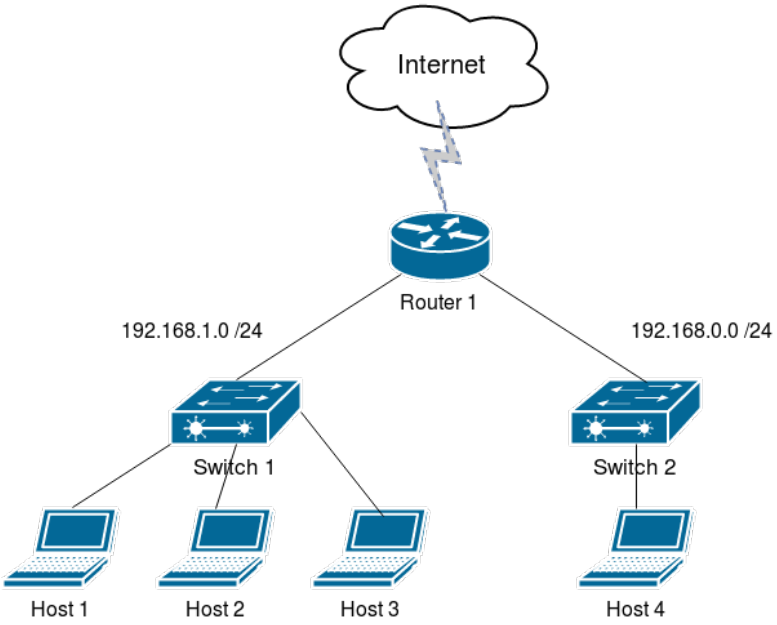

Figure 2: Topology 1

Mininet's name	Interface	IP address	MAC address
h1	h1-eth0	192.168.1.2/24	
h2	h2-eth0		
h3	h3-eth0		
h4	h4-eth0		
r1	r1-eth0		
r1	r1-eth1		

Table 3: Interface names and layer 2 and 3 addresses for the working topology.

```
git clone https://redmine.telecom-bretagne.eu/git/net_labs
cd net_labs
```

3) Go to the arp directory, for that run: `cd arp`

4) Verify the existence of the files called `topo_arp.py`, `createOVSswitchStandAlone.sh` and `killOVSswitch.sh`.

5) Verify that you have execution permissions for the `.sh` files, these scripts are going to be called by the python script. For that run `ls -lh` and verify the presence of an `x` at the first column of the output.

6) Execute the python script to emulate the network `sudo python3 topo_arp.py`. This will build up the topology and open a mininet console (`mininet>`)

4.1.2. Observing the content of the ARP cache

1) In host 1's console, type the following command `arp -n` or equivalently comand `ip neigh show`

Question 4.2.
Is there any content in h1's ARP cache? Do you find this normal? Explain. normal?

2) Perform a ping from h1 to h2, make sure to use the `-c` option to send only one packet.

3) Check again the content of h1's ARP cache with your preferred command

Question 4.3.
Do you observe any difference with respect to its previous content? Explain.

4) We will now try a ping to h4. From h1 ping h4

Question 4.4.
Do you expect any changes in the ARP cache with respect to its previous content?

5) Verify your answer to the previous question by looking at h1's ARP cache
(`arp -n` or `ip neigh show`)

Question 4.5.
Does the result match with what you expected? Explain each entry in the table.

6) Verify the content of r1's ARP cache

4.1.3. Observing ARP messages

We are now going to see the protocol in detail, understanding how the information that appears in the ARP cache is obtained. For that we are going to observe the exchanged ARP messages under different scenarios. First of all, let's start all over again from scratch, in order to make sure of the departure state. This means:

1) Put down mininet (`mininet> exit`)

2) Clean up mininet (`sudo mn -c`)

3) Run the topology again (`sudo python3 topo_arp.py`)

4) The information of all MAC addresses might be useful for the remaining of the lab. Find out the MAC addresses of all the relevant interfaces. You can use any of the two options seen before in the Introduction to Mininet lab, either opening a terminal in the component either typing the command directly from mininet's console. Please note that components names are only available within mininet. Complete Table 3 with the correct information.

5) Start three instances of wireshark, one in h1, another one h2 and finally h3. For that:

1. From the mininet console type `hi wireshark &` for i=1 to 3.
2. Start a capture at the `hi-eth0` for each wireshark. These interfaces correspond to the interfaces connected to the switch. (see Fig. 2).

6) Perform a one-time ping from h1 to h2

7) Observe the capture at h1's eth0. Use a visualization filter to see only ARP packets.

Question 4.6.
What is the protocol stack being used?

Question 4.7.
Which ARP messages do you see? Focus on the first two messages. For each of those ARP messages specify the content of the different fields. Deduce from that the purpose of the message. You can get help from the information provided in the Introduction (see section 3.1.1). In particular, pay attention to the origin and destination. Note: you can see some unexpected behavior on the messages ordering due to how wireshark handles the capture. If ever you notice this, discuss it with the instructor.

Question 4.8.
Look at the Ethernet frames carrying each of the previous two ARP messages and indicate for each of them the value of the type field, the source, and the destination fields. You have probably noticed that some of this messages are addressed to a broadcast address while others to a unicast one. We should see more details on this on Question 20.

Question 4.9.
Sender and receiver Hardware addresses (MAC addresses) are present in the Ethernet frame as well as in the ARP message carried by the Ethernet frame. Why do you think this information is duplicated?

8) Look now at the packets captured at h2-eth0.

Question 4.10.
Which ARP messages do you see?

Question 4.11.
Explore the ICMP messages, which is the protocol stack in use?

Question 4.12.
Which is the value of the field type in the Ethernet frame carrying an ICMP message?

9) Look at the packets captured at h3-eth0

Question 4.13.
Which ARP messages do you see? Explain.

Question 4.14.
What can you conclude about the functioning of ARP according to the previous captures? Explain the functioning of the protocol.

We are now going to analyse a situation where h1 communicates with a host in another LAN, in this case, h4.

Question 4.15.
What is the content of h1's arp cache?

1) From mininet console, open a terminal in h4 (`xterm h4`) once in h4, run a wireshark instance.

2) You can close the wireshark instances opened at h2 and h3

3) Start a new capture at h1-eth0 and a new capture at h4-eth0

4) Perform now a ping from h1 to h4.

5) Observe the capture **at h1**

Question 4.16.
Do you see any ARP message concerning this ping? Which ones? Detail.

6) Observe the first ICMP message (Echo request)

Question 4.17.
Observe the IP header, which is the destination IP? Observe the Ethernet frame, which is the destination MAC address? Explain.

7) Observe the capture **at h4**

Question 4.18.
Which ARP messages do you see? Pay attention to the involved source and destination addresses.

8) Observe the first ICMP message (Echo request)

Question 4.19.
Observe the IP header, which is the source IP? Observe the Ethernet frame, which is the source MAC address? Explain.

We are now going to explore what happens when a machine updates the content of its cache.

1. Use command `ip neigh show` to check the content of h1's ARP cache.
2. Find an entry whose state is STALE. If no entry is at this state, wait some seconds, and repeat previous and this step
3. Initiate a new wireshark capture in h1's eth0, use a visualization filter for ARP messages (type arp in the Filter field)
4. Perform a one time ping from h1 to the destination you identified as in state STALE in step 2
5. Analyse your capture

Question 4.20.
Which ARP messages do you see? In particular, observe the source and destination addresses, do you see any difference with what you have observed in question 7 in this subsection? Explain.

4.2. Netmasks

The ARP section is now closed. Let's open a new section. We are now going to turn our attention to the understanding of netmasks. You will be performing some troubleshooting over two different cases of interface misconfiguration in h1.

First case study

Suppose that interface h1-eth0 is configured with the same IP address but its netmask is set to /30 (while previously it was /24). We are going to set up this scenario and explore what happens with the connectivity between the different hosts.

1) Set h1's h1-eth0 to 192.168.1.2/30. For that you will use the ifconfig command, in h1's console type: `ifconfig h1-eth0 192.168.1.2/30`.

2) Start a wireshark capture at h1-eth0, h3-eth0 and r1-eth0.

3) Run a ping from h1 to r1.

Question 4.21.
The ping should be successful. Explain why. It could be useful for building your answer to look at the content of Table 2 and complete a line with the current case study. In particular think of the number of bits available for coding your host-part of the address. A handy command line tool is `ipcalc`.

4) Run a ping from h1 to h3.

Question 4.22.
The ping should not be successful. Write down the output of the ping and explain what happened. To understand what happend, you can use another linux command useful in this case, the `route` command, which shows the routing table of the host. Use it with the `-n` option to avoid name resolution.

5) Run a ping from h3 to h1

Question 4.23.
The ping should not be successful but you should obtain an output different to the previous one. Explain. Use the wireshark captures to support your explanation.

Second case study

Now we are going to explore another case. For that you are going to set the netmask of h1-eth0 to /23

1) Configure h1-eth0 using the command `ifconfig h1-eth0 192.168.1.2/23`.

2) Complete Table 2 with the current case study. Once again you might want to use ipcalc command line tool.

3) Run a ping with one packet option from h1 to h3

Question 4.24.
In this case the ping should be successful. Explain why.

4) Start a new wireshark capture in h1-eth0

5) Now, run a ping from h1 to h4

Question 4.25.
In this case, the ping should not be successful. Explain why. Support your answer explaining the contents of the capture.

5. Conclusion

Question 5.1.
What is the purpose of the ARP protocol?

Question 5.2.
Why do we have different addresses at layer 2 and layer 3?

Question 5.3.
Explain the purpose of addresses netmasks and how a network device uses them.

References

[1] RFC 826 - An Ethernet Address Resolution Protocol, or, Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, IETF, [online] https://tools.ietf.org/html/rfc826.

[2] RFC 4861 - Neighbor Discovery for IP version 6 (IPv6), IETF, [online] https://tools.ietf.org/html/rfc4861.

[3] Computer Networks, Fifth Edition Andrew S. Tanenbaum, David J. Wetherall, Prentice Hall

[4] Mininet, An Instant Virtual Network on your Laptop (or other PC) [online] https://mininet.org