Universidad de León
Bachelor Degree on Computer Engineering
Course on Computer Networks

Practical on the ARP protocol

Ancillary documentation for this practical

- *1. Skim-read the following presentation about ARP:*

  http://paloalto.unileon.es/cn/labs/CN-IP-ARP-ICMP-DHCP.pdf

- *2. The following document contains technical information about the ARP protocol that can be used as an introduction to ARP:*

  http://paloalto.unileon.es/cn/labs/CN-IP-ARP-ICMP.pdf

- *3. Decoding of the tcpdump trace to an ARP transaction (Request and Reply) is explained in this complementary note:*

  http://paloalto.unileon.es/cn/notes/arp-packet.pdf

Exercises for practice

Notes:

- *The MAC and IP addresses contained in the exercises to this script are illustrative, clarifying examples. In general, those addresses will not be coincident with the actual addresses appearing in the exercises as you solve them in Lab B6.*
- *Exercises #13 to the last one are not to be included in the LabBook as they are suggested for independent study.*

1. If you *are not* located at Lab B6, connect to your account in paloalto.unileon.es (ssh -p <port> …). If you are in Lab B6, then move forward to exercise no. 2.

```
$ ssh -p <port> <login name>@paloalto.unileon.es
```

a. *Hop* on two other hosts from Lab B6 after logging into paloalto.unileon.es. Select the two hosts from the *list of hosts* from Lab B6 net and check connectivity with them by using ping as in the following command sequence. If necessary,

power them up remotely by using the magic program that we developed in past practices:

```
$ wget http://paloalto.unileon.es/cn/Q/mac-ip.txt

$ /home/magic eno1 e0:d5:5e:d8:84:b6
```

b. Check that the host just powered up is accessible after waiting for 1 minute for its boot process to end:

```
$ ping 192.168.1.109
```

c. Do the same with another MAC address/IP address from the list, for example, use e0:d5:5e:dd:ed:2a which is allocated to IP address 192.168.1.141:

```
$ /home/magic eno1 e0:d5:5e:d8:84:b6
```

d. Now, check that the host is accessible after waiting for about 1 minute for its boot process to end:

```
$ ping -c 1 192.168.1.141
```

2. If you indeed are located in Lab B6, then, logon to one of the Lab computers (Hereafter referred to as Host H_A), and then request two terminals. Using one of the terminals, *hop* on to another host in Lab B6 (Hereafter referred to as H_B) by opening a remote ssh session to it:

```
$ ssh administrator@192.168.1.141
```

Check IP connectivity to H_B by using ping to its IP address, which we assume that is 192.168.1.141 in this illustration -any other host IP address from Lab B6 can be used, as long as that host is powered up. Limit the sent ICMP Echo Requests to 1 so that the sent and received traffic is the minimum, thereby avoiding clutter on your terminal:

```
$ ping -c 192.168.1.141
```

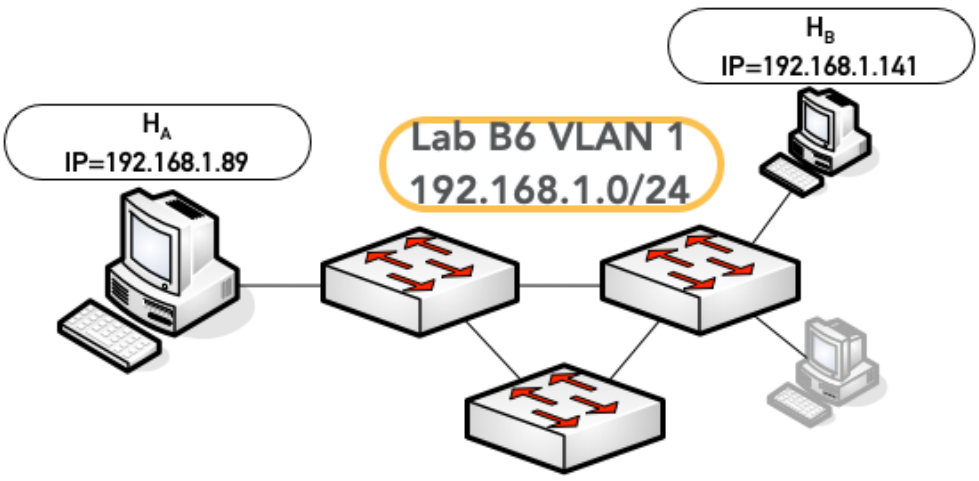

Figure 1. Lab B6 LAN with a *generalized* ARP Solicitor (H_A) and the responder ARP Neighbor (H_B) hosts

3. By this time, you should have one active ssh session, regardless of your location, local or remote. Herein, we use the convention of calling each of the session hosts H_A and H_B, respectively. In all of them, use the habitual *administrator* user and the password that is written on the upper, left board corner for switching to super-user (su command by default in Debian).

At the ssh session to H_B, obtain a listing of configured network interfaces. Observe the *concrete* MAC address to the NIC the ssh connection is sent over, for example, in the ifconfig listing that follows, the MAC address is highlighted in red:

```
$ ifconfig

en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
 options=50b<RXCSUM,TXCSUM,VLAN_HWTAGGING,AV,CHANNEL_IO>
 ether b0:e5:f9:f1:fe:41
 inet6 fe80::10bf:8402:ac90:7755%en0 prefixlen 64 secured scopeid 0x6
 inet6 fdfb:bdec:783e:0:8f:c120:587c:d121 prefixlen 64 autoconf secured
 inet 192.168.1.141 netmask 0xffffff00 broadcast 192.168.1.255
 nd6 options=201<PERFORMNUD,DAD>
 media: autoselect (1000baseT <full-duplex,flow-control,energy-efficient-
ethernet>)
 status: active
```

At this moment, after H_A has created a successful connection to H_B. That MAC address *is represented as* H_A. It has been *transparently* recorded in the ARP table to host H_A, that is, without your becoming aware of that recording whatsoever.

Finish this step by writing down or by copying the MAC address of the NIC used by H_B so you can check it later.

4. (Until otherwise indicated in the text, do the exercises that follow, in H_A). Get the current contents of H_a arp table by submitting the following arp command:

```
$ arp -a
```

It may be necessary that you wait a little while before the listing of IP to MAC mappings is output. That listing represents the local mappings of IP-to-MAC that have been *learned* by your host (H_A) as it needed them for local network communication. As we said above, at this time H_A *knows* the MAC address to the NIC of H_B. In the arp table listing, an entry for 192.168.1.141 should appear. That is normal: H_A wanted it to contact 192.168.1.141 (For ssh and for ping, at least, so far), then ARP resolved it and stored it in the ARP table.

5. At this time, we want to observe the operation of the ARP protocol. To that end, for you to be able to observe the ARP transactions as they unfold, it is necessary that you do this, in order:

    a. Close your ssh session on H_b!

    b. Clear host H_a arp table entry corresponding to host H_b. If you remove that entry, then you'll be able to watch what ARP does in order to find out the MAC address to H_b. Execute these commands at H_a:

```
$ su
Password:

# arp -d 192.168.1.141

# arp -a
… listing of IP-to-MAC mappings currently cached at HA
```

The preceding command should print no mapping to IP address 192.168.1.141, at this time, since we removed it from the arp table, above.

6. *Check again* the current contents of the arp table of host H_A. Issuing the arp -a command is of help since habitually it has to be issued repeatedly until the kernel fulfils the request to clear an arp entry. Anew, note the listing should contain no mapping for H_b:

```
$ arp -a
… listing of IP-to-MAC mappings currently cached at Hb
… this listing should contain no mapping for the IP address of
host Hb
```

7. Finally, host H_a *shouldn't have* any mapping of the MAC address corresponding to H_b. *If you send a* ***ping*** *to* H_b, now, H_a host will have to find out the MAC address to H_b *ever before sending the first ICMP* ***Echo Request*** *packet* (Resulting from your ping command). Resolving MAC addresses given their IP address is the **job of ARP!**

    In summary before actually sending the first IP packet encapsulating your ping's ICMP ECHO REQUEST, your host (H_a) will send an ARP REQUEST to the broadcast address; shortly afterwards, host H_b should respond with an ARP REPLY carrying the requested MAC address (That of H_b) encapsulated into an Ethernet frame (A Unicast communication, in this case). You may want to see the explanation about the messages involved in an ARP transaction by skimming the document pointed to by the 3rd document included in Ancillary Documentation section, above.

The diagram in Fig. 2 summarizes the preceding ARP transaction: The sending of an ARP Request (Broadcast) and the sending back of the corresponding ARP Reply (Unicast).

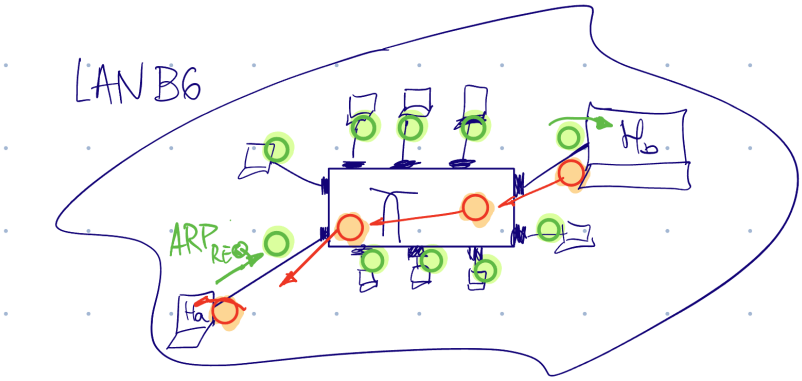

Figure 2. Overall diagram to a *basic* ARP transaction for the resolution of the MAC address to host H_b. In green, the request; in red, the response.

8. Observe the arp messages resulting in an the arp resolution. To check that the TCP/IP stack at host H_A sends the ARP Request to the broadcast address, run tcpdump at one of the terminals that you started earlier at H_A. Use the following command. Make sure that you select the right NIC from H_A, which is usually eno1 in many hosts in Lab B6:

```
# whereis ifconfig
/usr/sbin/ifconfig

# PATH=$PATH:/usr/sbin:.

# ifconfig
(Select the main NIC from the listing, for example, eno1)

# tcpdump -i eno1 -entx -XX -vvv arp or icmp
```

9. At the other local terminal that you started earlier in H_A, send *one* ICMP Echo Request to host H_B (Use the ping command below). Observe the trace captured at the other terminal in H_A which should contain four frames, as we explained above:

```
$ ping -c 1 192.168.1.141
```

Frames captured at H_A and observed with tcpdump:

1st. ARP Request for H_B sent by H_A (Broadcast mode)

2nd. ARP Reply sent by H_B and received by H_A (Unicast mode)

3rd. ICMP Echo Request from H_A to H_B (Unicast)

4th. ICMP Echo Reply from H_B to H_A (Unicast)

10. With the tcpdump trace obtained at H_A, check that Host H_B reacted to the ARP Request by sending back the corresponding ARP Reply. Highlight the MAC address included by H_B in its response ARP packet. It should be the same MAC address that you obtained in the ifconfig listing earlier at H_B, above, at step 3 (`b0:e5:f9:f1:fe:41`).

11. Check the contents of the arp table to host H_A. The mapping to host H_B have been created there at this time. Entries on this table are removed automatically by the *Linux neighboring system* after some number of minutes have elapsed, after which those entries are assumed to become stale. After removal, a new protocol transaction will have to be done by the stack if that IP-to-MAC mapping is needed again.

12. **Document and explain** the results that you have obtained, as much in H_A as in H_B. If necessary, repeat all the steps if some results are not what you expected, or some unexpected interaction with other class mates ARP transactions took place; maybe you want to change the *experiment* somehow so that you better understand certain aspects of this practical, or the relevant lectures. Check out the ancillary documentation pointers included in the heading of this document.

The exercises that come below have a conceptual character and are suggested for independent study, in connection with Internet architecture and protocol stacks. Their inclusion in the LabBook for academic year 2024 is not mandatory.

13. Obtain the protocol stack to an ARP Request with all the relevant multiplexing keys (See fig. 3).

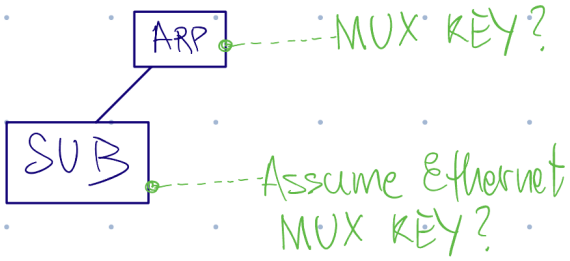

Figure 3. ARP protocol stack

14. Obtain as well, the protocol stack to an ICMP Echo Request with all the relevant multiplexing keys (See fig. 4).

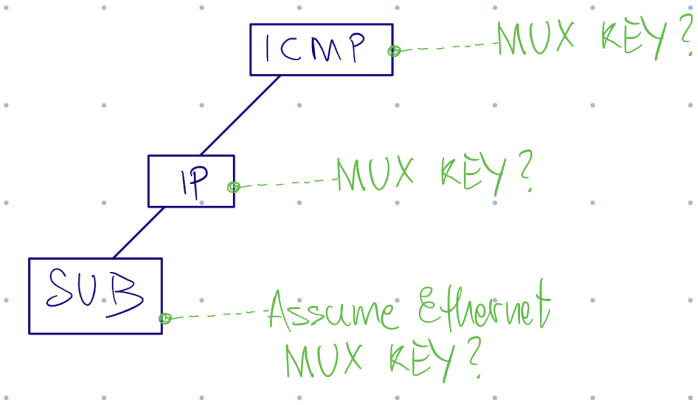

Figure 4. ICMP protocol stack

15. What is the *architectural* relationship between ARP and IP in the protocol stack in Fig. 5? Is it the same as the relationship between IP and SUB? Or, maybe is it the relationship between IP and ICMP?

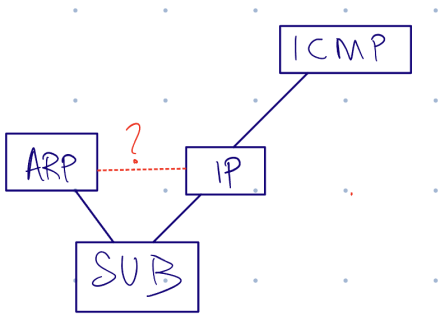

Figure 5. Partial Internet protocol stack implemented in today's Linux

16. Explain why some ARP requests are sent by using the *unicast* communication mode instead of the *broadcast* one.

17. What is *gratuitous ARP?* Explain it briefly. If necessary, consult the Ancillary documentation at the beginning of this document. You might also access the RFC to the DHCP protocol).

18. *Create* a scenario to observe Gratuitous ARP after a DHCP client retrieves its IP address. The following outline is a reference, it is thereby i*ncomplete.*

a. Configure one of the secondary NICs to host H_B to have their IP address be delivered by the local net DHCP server. For example, assuming that H_B has a secondary NIC which label is enp1s0, you have to include the following line in /etc/network/interfaces:

```
auto enp1s0
iface enp1s0 inet dhcp
```

b. Power off H_B:

```
# shutdown -r now
```

c. Start tcpdump with appropriate options to observe the packets involved in Gratuitous ARP

d. Send the magic packet to H_B, or press its power up button

e. Observe the gratuitous ARP messages at H_A after H_B boots up and retrieves its IP address from the DHCP server running in 192.168.1.1. Reflect on this experiment extensively and modify it as necessary so that it results in a repeatable experiment, one that will consistent results in general circumstances.