ANNEX A · ETHERNET, MAC AND SWITCHING

Name: _____ Group G = _____ Date: _____

TASK 1 **Your real network card (on the Ubuntu host, not in the lab)**

```
ip -br link
```

Name of the wired card: _____ Its MAC address: _____

Type the first three bytes into `https://www.wireshark.org/tools/oui-lookup.html`. Manufacturer:

_____

DONE WHEN you have a maker's name, and you compared your first three bytes with a neighbour's.

TASK 2 **A network card in the lab**

In the folder `~/labs` (use your group number):

```
G=3 containerlab deploy -t lan.clab.yml
docker exec -it clab-lan-pc1 bash
ip link show eth1
```

MAC of pc1 (`link/ether`): _____ `brd`: _____

`mtu`: _____ `state`: _____ Second hex digit of the MAC: ____ → made by a factory / made up by software (circle)

DONE WHEN every blank is filled and you can say what `brd` means.

TASK 3 **Be the switch (pairs)**

A switch with four ports: PC **A** on port 1, **B** on port 2, **C** on port 3, **D** on port 4. Its table is empty. Six frames arrive in this order.

#	Frame	The switch learns (MAC → port)	The switch sends it to port(s)
1	A → B		
2	B → A		
3	C → A		
4	D → `ff:ff:ff:ff:ff:ff`		
5	A → D		
6	B → E (E is switched off)		

DONE WHEN the final table has four entries and you know which two frames were flooded for different reasons.

TASK 4 Watch ARP ask and answer

Open a second terminal for **pc2**. Then, in this order:

```
ip neigh                                    # on pc1: what do you see?
tcpdump -e -n -c 4 -i eth1 arp or icmp      # on pc2 (it waits for 4 frames)
ping -c 3 10.G.0.12                         # on pc1
ip neigh                                    # on pc1, again
```

First `ip neigh`: _____ Second `ip neigh`, the word at the end: _____

Frame 1 is sent to the MAC _____ and asks: who-has _____ tell _____

Frame 2 answers: _____ is-at _____ Frames 3 and 4 are: _____

DONE WHEN the MAC in the answer of frame 2 is the same one that `ip neigh` shows on pc1.

TASK 5 Watch the switch learn

Open a third terminal: `docker exec -it clab-lan-sw bash`. The switch's table is the **fdb**.

```
bridge fdb flush dev br0                    # on sw: forget everything
bridge fdb show br br0 | grep -v permanent  # on sw: empty?
ping -c 1 10.G.0.12                         # on pc1
bridge fdb show br br0 | grep -v permanent  # on sw: who is there now?
```

Lines after the ping: _____ The MAC of pc1 is on port _____ The MAC of pc2 is on port _____

Why is pc3 not in the table? _____

What must you do so that pc3 appears? Do it. Command: _____

DONE WHEN the table shows three MAC addresses, one per port `eth1`, `eth2`, `eth3`.

TASK 6 A ping to nobody

```
ping -c 2 -W 1 10.G.0.99                    # on pc1: no machine has this address
ip neigh                                    # on pc1
```

State of 10.G.0.99 in `ip neigh`: _____ Did an ICMP echo request ever leave pc1? _____

Why? _____

DONE WHEN you can explain it with the word ARP. Then clean up: `exit` in all terminals and `containerlab destroy -t lan.clab.yml --cleanup`.

EXIT TICKET Circle one answer. Alone, no notes.

1. A switch decides where to send a frame by looking at...
a) the source IP address **b)** the destination IP address
c) the destination MAC address **d)** the TTL

2. A switch learns which MAC is behind which port from...
a) the source MAC of the frames that arrive **b)** the destination MAC of the frames that arrive
c) a list typed by the administrator **d)** the DHCP server

3. A frame arrives for a MAC that is not in the table. The switch...
a) throws it away **b)** sends an error to the sender
c) asks the router **d)** sends it out of every port except the one it came in

4. Which question does ARP answer?
a) Which IP address has this name? **b)** Which MAC address has this IP address?
c) Which port is this MAC on? **d)** Which router leads to this network?

5. The destination MAC of an ARP request is...
a) `ff:ff:ff:ff:ff:ff` **b)** `00:00:00:00:00:00`
c) the MAC of the switch **d)** the MAC of the router