



---

## ΕΡΓΑΣΤΗΡΙΟ ΔΙΚΤΥΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

---

### ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3: ΤΟΠΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΜΕΤΑΓΩΓΕΙΣ LAN



22 ΜΑΡΤΙΟΥ, 2023

ΘΟΔΩΡΗΣ ΑΡΑΠΗΣ – EL18028

Όνοματεπώνυμο: Θοδωρής Αράπης	Ομάδα: 3
Όνομα PC/ΛΣ: DESKTOP-JGHL94V/ WINDOWS 10	Ημερομηνία: 22/3/2023

## Άσκηση 1: Γέφυρα – Διασύνδεση 2 LAN

### 1.1

Με τις εντολές “**ifconfig em0 192.168.1.1/24**” και “**ifconfig em0 192.168.1.2/24**” στα PC1 και PC2 αντίστοιχα.

### 1.2

Με τις εντολές “**ifconfig em0 up**” και “**ifconfig em1 up**” ενεργοποιούμε τις διεπαφές em0 και em1.

### 1.3

Για διευκόλυνση τροποποιούμε κατάλληλα τα αρχεία /etc/hosts προσθέτοντας στο μεν PC1 τη γραμμή “192.168.1.2 PC2 PC”, ενώ στο PC2 τη γραμμή “192.168.1.1 PC1 PC1”. Επιπλέον, αλλάζουμε από το VirtualBox τα δίκτυα των PC1 και PC2 σε LAN1 και LAN2 για να ‘ναι σύμφωνα με την τοπολογία του δικτύου. Εκτελώντας, επομένως, τα κατάλληλα ping, παρατηρούμε από αμφότερα τα μηχανήματα πως τα πακέτα που έστειλαν απλά χάθηκαν.

### 1.4

Παρατηρούμε πως δε παράγονται ICMP πακέτα, παρά μόνο ARP πακέτα με τα οποία το PC1/PC2 ρωτά ποιος έχει τη διεύθυνση του PC2/PC1. Ο λόγος που συμβαίνει αυτό είναι πως οι υπολογιστές δεν έχουν κάποια καταχώρηση στον ARP πίνακα ώστε να ξέρουν σε ποιον ανήκει η εκάστοτε IP, αφού δεν έχουν επικοινωνήσει έως τώρα.

```
root@PC:~ # tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on em0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:28:27.921995 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 46
17:28:28.991360 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 46
17:28:30.042950 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 46
17:28:31.093326 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 46
```

(ping από το PC1 στο PC2 και tcpdump στην em0 που ανήκει στο LAN1).

### 1.5

Αρχικά δημιουργούμε τη γέφυρα με την εντολή **"ifconfig bridge0 up"**. Στη συνέχεια, προσθέτουμε τις διεπαφές em0 και em1 και ενεργοποιούμε την γέφυρα με την εντολή **"ifconfig bridge0 addm em0 addm em1 up"**.

### 1.6

Αυτή τη φορά, τα ping επιτυγχάνουν κανονικά.

```
root@B1:~ # tcpdump -i bridge0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on bridge0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:43:48.336966 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 17923, seq 0, length 64
17:43:48.337223 IP 192.168.1.1 > 192.168.1.2: ICMP echo reply, id 17923, seq 0, length 64
17:43:49.386663 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 17923, seq 1, length 64
17:43:49.386926 IP 192.168.1.1 > 192.168.1.2: ICMP echo reply, id 17923, seq 1, length 64
17:43:54.641751 IP 192.168.1.1 > 192.168.1.2: ICMP echo request, id 16899, seq 0, length 64
17:43:54.641935 IP 192.168.1.2 > 192.168.1.1: ICMP echo reply, id 16899, seq 0, length 64
17:43:55.694499 IP 192.168.1.1 > 192.168.1.2: ICMP echo request, id 16899, seq 1, length 64
17:43:55.694731 IP 192.168.1.2 > 192.168.1.1: ICMP echo reply, id 16899, seq 1, length 64
```

### 1.7

Παρατηρούμε πως το πεδίο TTL έχει τιμή 64 που είναι και η default, επομένως είναι σαν το PC2 και το PC1 να συνδέονται άμεσα, πράγμα λογικό αφού η γέφυρα είναι "διαφανής" για το δίκτυο.

### 1.8

Παρατηρούμε τα παρακάτω αποτελέσματα στο PC1 και στο PC2 αντίστοιχα:

```
root@PC1:~ # arp -a
? (192.168.1.1) at 08:00:27:f0:ee:f2 on em0 permanent [ethernet]
? (192.168.1.2) at 08:00:27:e4:eb:5a on em0 expires in 756 seconds [ethernet]
```

```
root@PC2:~ # arp -a
PC1 (192.168.1.1) at 08:00:27:f0:ee:f2 on em0 expires in 768 seconds [ethernet]
? (192.168.1.2) at 08:00:27:e4:eb:5a on em0 permanent [ethernet]
```

Έχουν γίνει επομένως οι κατάλληλες αντιστοιχίσεις IP με MAC διευθύνσεις, χωρίς να παρεμβάλλονται MAC της γέφυρας.

## 1.9

Από την αρχική κονσόλα εκτελούμε **“tcpdump -i em0 -ev”** για να λαμβάνουμε κίνηση της διεπαφής που είναι στο LAN1 (em0), ενώ σε δεύτερη κονσόλα χρησιμοποιούμε την εντολή **“tcpdump -i em1 -ev”** για να λαμβάνουμε κίνηση της διεπαφής που είναι στο LAN2 (em1). Για παράδειγμα, από την πρώτη κονσόλα λαμβάνουμε το παρακάτω (ακριβώς το ίδιο λαμβάνουμε και στη δεύτερη, αλλά με διαφορετικούς χρόνους σύλληψης):

```
root@PC:~ # tcpdump -i em0 -ev
tcpdump: listening on em0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:58:55.336448 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:e4:eb:5a (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 54670, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.1 > 192.168.1.2: ICMP echo request, id 23811, seq 0, length 64
17:58:55.336710 08:00:27:e4:eb:5a (oui Unknown) > 08:00:27:f0:ee:f2 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 45422, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.2 > 192.168.1.1: ICMP echo reply, id 23811, seq 0, length 64
17:58:56.377723 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:e4:eb:5a (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 54671, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.1 > 192.168.1.2: ICMP echo request, id 23811, seq 1, length 64
17:58:56.377969 08:00:27:e4:eb:5a (oui Unknown) > 08:00:27:f0:ee:f2 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 45423, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.1.2 > 192.168.1.1: ICMP echo reply, id 23811, seq 1, length 64
```

Αυτό που παρατηρούμε, είναι πως το request πακέτο φεύγει από την MAC 08:00:27:f0:ee:f2 (PC1) προς την 08:00:27:e4:eb:5a (PC2), ενώ το reply κάνει την αντίστροφη πορεία.

## 1.10

Όχι, δε φαίνεται κάποια αλλαγή μεταξύ των 2 καταγραφών.

## 1.11

Δεν αλλάζει κανένα πεδίο.

## 1.12

Δεν υπάρχει καμία ένδειξη της γέφυρας, όπως φαίνεται παρακάτω. Όπως έχουμε πει, για το δίκτυο, η γέφυρα είναι “διαφανής”.

```
root@PC1:~ # traceroute PC2
traceroute to PC2 (192.168.1.2), 64 hops max, 40 byte packets
 1  PC2 (192.168.1.2)  1.016 ms  0.318 ms  0.315 ms
```

### 1.13

Εκτελούμε στο B1 την εντολή **"tcpdump -vvve -i em1"** και κάνουμε **"ping -c 10 PC2"** από το PC1.

### 1.14

Με την αλλαγή της IP, η γέφυρα σταματά να προωθεί τα πακέτα προς το PC2. (Φαίνεται να τα προωθεί στην προηγούμενη διεύθυνση ip το PC2)

### 1.15

Το ping πλέον στέλνει κανονικά τα requests του, αλλά δε λαμβάνει κανένα reply.

### 1.16

Το ping αποτυγχάνει με το μήνυμα **"host is down"**. Αυτό συμβαίνει, καθώς δεν υπάρχει κάποια IP στο LAN3 που να αντιστοιχεί σε αυτήν που στέλνει το πακέτο το PC3.

### 1.17

Εκτελούμε **"ifconfig em2 up"** και μετά **"ifconfig bridge0 addm em2 up"**.

### 1.18

Πλέον λαμβάνουμε κανονικά απάντηση.

### 1.19

Δε καταγράφεται κανένα ICMP πακέτο στην em1 όταν κάνουμε ping από το PC1 στο PC3 ή αντίστροφα. Ο λόγος είναι πως η κάρτα αυτή ούτε παίζει ρόλο στη δρομολόγηση των πακέτων μεταξύ των 1 και 3, αλλά ούτε και ανήκει σε κοινό LAN με κάποιο από τα 2 PC, ώστε να ανιχνεύσει κίνηση λόγω της επιλογής **"Allow VMs"**.

### 1.20

Αυτή τη φορά, καταγράφηκε το εξής παρακάτω ARP request. Ο λόγος που έγινε αυτό, είναι πως αφού το PC1 δεν ήξερε την MAC διεύθυνση που αντιστοιχεί στην IP του PC3, το πακέτο προωθήθηκε και στο LAN2 αλλά και στο LAN3 (broadcast) προκειμένου να το απαντήσει ο υπολογιστής με IP αυτή του PC3 και να μάθουμε την MAC address του, ώστε να επιτευχθεί η δρομολόγηση του πακέτου.

```
root@PC:~ # tcpdump -i em1 -ev
tcpdump: listening on em1, link-type EN10MB (Ethernet), capture size 262144 bytes
18:42:56.192796 08:00:27:f0:ee:f2 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.3 tell 192.168.1.1, length 46
```

### 1.21

Με την εντολή “**ifconfig bridge0**”.

### 1.22

Με την εντολή “**ifconfig bridge0 addr**”.

```
root@B1:~ # ifconfig bridge0 addr
08:00:27:e4:eb:5a Vlan1 em1 1193 flags=0<>
08:00:27:6b:3b:db Vlan1 em2 548 flags=0<>
08:00:27:f0:ee:f2 Vlan1 em0 1193 flags=0<>
```

### 1.23

Η πρώτη διεύθυνση αντιστοιχεί στο PC2, η δεύτερη στο PC3 και η τρίτη στο PC1.

### 1.24

Εκτελούμε “**ifconfig bridge0 flush**”.

### 1.25

Εκτελούμε “**ifconfig bridge0 deletem em2**”.

### 1.26

Εκτελούμε “**ifconfig bridge0 destroy**”.

### 1.27

Σε καθένα από τα PC1, PC2, PC3 εκτελούμε “**ifconfig em0 delete**”.

## Άσκηση 2: Αυτό-εκπαίδευση γεφυρών

### 2.1

Προτού ορίσουμε τις διευθύνσεις, κάνουμε τις κατάλληλες αλλαγές στο hostname (στο αρχείο /etc/rc.conf) και στις αντιστοιχίσεις ονομάτων – IP (στο αρχείο /etc/hosts). Για να ορίσουμε διευθύνσεις, εκτελούμε σε κάθε PC την εντολή “**ifconfig em0 192.168.1.X/24**” όπου **X** = 1, 2, 3, 4 ανάλογα το PC.

## 2.2

Στο μηχάνημα B1, εκτελούμε διαδοχικά τις εντολές **“ifconfig em0 up”**, **“ifconfig em1 up”**, **“ifconfig bridge1 create”** και **“ifconfig bridge1 addm em0 addm em1 up”**.

## 2.3

Στο μηχάνημα B2, εκτελούμε διαδοχικά τις εντολές **“ifconfig em0 up”**, **“ifconfig em1 up”**, **“ifconfig bridge2 create”** και **“ifconfig bridge2 addm em0 addm em1 up”**.

## 2.4

Στο μηχάνημα B3, εκτελούμε διαδοχικά τις εντολές **“ifconfig em0 up”**, **“ifconfig em1 up”**, **“ifconfig bridge3 create”** και **“ifconfig bridge3 addm em0 addm em1 up”**.

## 2.5

Εκτελούμε την εντολή **“ifconfig em0”** σε κάθε PC και βρίσκουμε τις MAC διευθύνσεις που αναγράφουμε παρακάτω. Επιπλέον, εκτελούμε **“arp -ad”**.

- PC1: 08:00:27:f0:ee:f2
- PC2: 08:00:27:e4:eb:5a
- PC3: 08:00:27:6b:3b:db
- PC4: 08:00:27:65:8f:a8

## 2.6

Σε κάθε γέφυρα B1, B2, B3 εκτελούμε **“ifconfig bridgeX flush”**, όπου **X = 1, 2, 3** αντίστοιχα.

## 2.7

Εκτελούμε **“tcpdump -i em0”** σε κάθε PC.

## 2.8

Παρουσιάζονται διαδοχικά οι πίνακες προώθησης για το B1, B2, B3:

```
root@B1:~ # ifconfig bridge1 addr
08:00:27:e4:eb:5a Vlan1 em1 1192 flags=0<>
08:00:27:f0:ee:f2 Vlan1 em0 1192 flags=0<>
```

```
root@B2:~ # ifconfig bridge2 addr
08:00:27:e4:eb:5a Vlan1 em0 1188 flags=0<>
08:00:27:f0:ee:f2 Vlan1 em0 1188 flags=0<>
```

```
root@B3:~ # ifconfig bridge3 addr
08:00:27:f0:ee:f2 Vlan1 em0 1177 flags=0<>
```

## **2.9**

Η γέφυρα B1 έχει καταχωρήσει την MAC του PC1 στην em0 (LAN1) και του PC2 στην em1 (LNK1). Η B2 έχει καταχωρήσει τις MAC των PC1 και PC2 στην em0 (LNK1). Η B3 έχει καταχωρήσει μόνο την MAC του PC1 στην em0 (LNK2). Οι καταχωρήσεις έγιναν ως εξής:

- Το PC1 έστειλε ARP request για να μάθει την MAC του PC2, οπότε και το B1 καταχώρησε την  $MAC_{PC1}$  στην em0 (LAN1).
- Από το B1, στάλθηκε μέσω της em1 στο LNK1 το ARP request, οπότε και το έλαβε ο στόχος PC2, αλλά και η B2, οπότε και το καταχώρησε στο em0 (LNK1).
- Το B2 με τη σειρά του, το προώθησε στο LNK2, οπότε και το έλαβε το PC3 αλλά και η B3, η οποία το καταχώρησε στο em0 (LNK2).
- Στο μεταξύ, το PC2 εξέπεμψε την απάντησή του στο LNK1, την οποία και έλαβε η B1 και η B2, καταχωρώντας έτσι την  $MAC_{PC2}$  η μεν B1 στο em1 (LNK1) και η δε B2 στο em0 (LNK1).

## **2.10**

Η επικοινωνία συνεχίζει να αφορά τους PC1 και PC2, ωστόσο αυτή τη φορά ο PC2 γνωρίζει από πριν την MAC του PC1. Επομένως, δε κάνει ARP request, με αποτέλεσμα το B3 να συνεχίζει να γνωρίζει μόνο την MAC του PC1. Προφανώς οι πίνακες των B1, B2 δεν αλλάζουν επίσης καθόλου, καθώς ήδη γνώριζαν για τα PC1 και PC2.

## **2.11**

Πλέον όλες οι γέφυρες έχουν την MAC του PC4 (και του PC2). Η καταγραφή αυτή υπάρχει στο B1, καθώς, προκειμένου το PC2 να στείλει το πακέτο στο PC4 έκανε αρχικά ένα ARP request. Όταν το PC4 το έλαβε, εξέπεμψε το ARP reply, το οποίο και ο B3 έκανε broadcast στο LNK2. Παραλαμβάνοντάς το από εκεί, επίσης broadcast στο LNK1 έκανε το B2 με τελικό αποτέλεσμα η καταγραφή  $MAC_{PC4}$  να υπάρχει και στο B1.

(Πιθανόν όχι από broadcast, αλλά από το Allow VM's, καθώς ο B3 πρέπει να το έστειλε αμέσως στο B2 και όχι να έκανε broadcast)

## **2.12**

Παρατηρούμε πως με το συγκεκριμένο ring προστέθηκε σε όλους τους πίνακες δρομολόγησης η MAC του PC3. Αυτό συνέβη, καθώς όπως βλέπουμε από τις καταγραφές, το PC3 έκανε αρχικά ένα ARP request για την MAC του PC2, επομένως γνωστοποίησε τη δική του MAC αρχικά στα B2, B3. Στη συνέχεια, το PC2 είναι στο κομμάτι LNK1 για το B2, επομένως προωθείται εκεί το request, όπου και λαμβάνεται από το B1, με αποτέλεσμα να ενημερώσει και εκείνο τις καταχωρήσεις του με την  $MAC_{PC3}$ . Ύστερα απαντάει το PC2



και δεν έχουμε καμία καινούρια ενημέρωση αφού ήδη περιέχεται η αντιστοίχιση για την MAC του PC2 σε όλες τις γέφυρες.

### **2.13**

Εκτελούμε “**ping PC2**” από τα PC1 και PC4 σε δεύτερη κονσόλα στο καθένα.

### **2.14**

Το ping από το PC4 στο PC2 συνεχίζει κανονικά.

### **2.15**

Αντιθέτως, το ping από το PC1 προς το PC2 συνεχίζει να στέλνει ICMP Echo requests χωρίς, ωστόσο, να λαμβάνει απάντηση. Αυτό που συμβαίνει είναι πως η γέφυρα B1 στέλνει συνέχεια τα πακέτα στο LNK1, εφόσον έχει αποθηκευμένη τη παλιά θέση του PC2 στο δίκτυο και δεν έχει ενημερωθεί για τη νέα, ενώ επίσης το B2 δε προωθεί τα πακέτα στο LNK2, διότι και για αυτό επίσης το PC2 είναι ακόμα στο LNK1. Η ενημέρωση θα γίνει μόλις το PC2 παράξει πλαίσιο.

### **2.16**

Παρατηρούμε πως πλέον το PC1 λαμβάνει κανονικά απάντηση ξανά από το PC2. Αυτό που συνέβη είναι πως το PC2 έστειλε πακέτο στο PC3, επομένως, στάλθηκε πακέτο στο B3, το οποίο επίσης μεταδόθηκε στο LNK2. Από εκεί, το B2 ενημέρωσε τον πίνακά του πως το PC2 βρίσκεται πλέον από τη θύρα em1 αντί για την em0 που ήταν πριν. Επομένως, από το αμέσως επόμενο πακέτο που έστειλε το PC1, αυτό έφτασε στο B1, στάλθηκε στο LNK1, αλλά από εκεί όταν παρελήφθη από το B2 προωθήθηκε στο LNK2 για να φτάσει τελικά στο PC2.

### **2.17**

Εκτελώντας “**ifconfig bridge1**” στο B1, βλέπουμε πως ο μέγιστος χρόνος που διατηρείται μια εγγραφή στον πίνακα προώθησης είναι 1200 seconds, άρα μετά από 20 λεπτά θα μάθαινε μόνη της η γέφυρα για την αλλαγή.

### Άσκηση 3: Καταιγίδα πλαισίων εκπομπής

Ορίσαμε:

#### B1:

**em0** → LAN1

**em1** → LNK1

**em2** → LNK2

#### B2:

**em0** → LNK1

**em1** → LNK2

**em2** → LAN2

#### 3.1

Έχοντας ήδη ενεργές τις διεπαφές em0 (LAN1), em1 (LNK1), εκτελούμε **“ifconfig bridge1 create”** και μετά **“ifconfig bridge1 addm em0 addm em1 up”**.

#### 3.2

Έχοντας ήδη ενεργές τις διεπαφές em0 (LNK1), em2 (LAN2), εκτελούμε **“ifconfig bridge2 create”** και μετά **“ifconfig bridge2 addm em0 addm em2 up”**

#### 3.3

Εκτελούμε την εντολή **“ifconfig em0”** σε κάθε PC και βρίσκουμε τις MAC διευθύνσεις που αναγράφουμε παρακάτω. Επιπλέον, εκτελούμε **“arp -ad”**.

- PC1: 08:00:27:f0:ee:f2
- PC2: 08:00:27:e4:eb:5a
- PC3: 08:00:27:6b:3b:db

#### 3.4

Παρατηρούμε πως καταγράφεται ARP κίνηση, όπου κάνοντας ping από το PC2 στο PC3, το PC2 ρωτάει για την MAC της IP PC3. Αναλυτικότερα, δεδομένου ότι το PC2 δε γνώριζε τη MAC του PC3 για να στείλει αμέσως το πακέτο, έστειλε ένα ARP request για τη μάθει. Δεδομένου ότι το PC3 βρίσκεται στο ίδιο LAN με αυτό, έμαθε αμέσως την απάντηση οπότε και στάλθηκε το ICMP Echo μήνυμα. Ωστόσο, παράλληλα, προωθήθηκε το ARP Request μέσω του B2 στο LNK1 και από εκεί στο B1, το οποίο και προώθησε το request τέλος στο PC1.

#### 3.5

Κάνουμε **“ping PC1”**.

### 3.6

Αφού ενεργοποιήσουμε την em2 και την em1 στα B1 και B2 αντίστοιχα, τα κάνουμε add στην εκάστοτε γέφυρα με την εντολή “**ifconfig bridgeX addm em2 up**”, όπου X = 1, 2 αντίστοιχα.

### 3.7

Με “**ifconfig bridgeX addr**” βλέπουμε τα παρακάτω για τα bridge1 και bridge2:

```
root@PC:~ # ifconfig bridge1 addr
08:00:27:f0:ee:f2 Vlan1 em0 1191 flags=0<>
08:00:27:6b:3b:db Vlan1 em1 1191 flags=0<>
08:00:27:e4:eb:5a Vlan1 em1 933 flags=0<>
```

```
root@PC:~ # ifconfig bridge2 addr
08:00:27:f0:ee:f2 Vlan1 em0 1181 flags=0<>
08:00:27:6b:3b:db Vlan1 em2 1181 flags=0<>
08:00:27:e4:eb:5a Vlan1 em2 923 flags=0<>
```

### 3.8

Στο B1 αλλά και στο B2, η MAC του PC1 εμφανίζεται στη διεπαφή em0, ενώ η MAC του PC3 στη διεπαφή em1 του B1 και στην διεπαφή em2 του B2.

### 3.9

Εκτελούμε “**tcpdump -i em0 -ev**”.

### 3.10

Το ping δεν επιτυγχάνει.

### 3.11

Αποσυνδέουμε τα καλώδια των LNK1 από τα B1, B2. Πλέον, το PC1 εμφανίζεται στην em1 του B2 και το PC3 εμφανίζεται στο em0. Αναλυτικά: το PC3 στέλνει το ARP request στο LAN2, το οποίο παραλαμβάνει το PC2 και το B2. Από εκεί, αυτό αποστέλλεται προς το LNK1 και το LNK2, επομένως, πλέον το B1 που τα παραλαμβάνει θεωρεί πως το PC3 είναι στο LNK1 ή στο LNK2 (όποιο παρέλαβε τελευταίο). Αφού προωθήσει το πακέτο στο LAN1, το προωθεί και στο LNK1 και LNK2, αφού το έλαβε από το LNK2 και LNK1. Επομένως, τα λαμβάνει ο B2 και θεωρεί πλέον πως το PC3 είναι στο em0 ή στο em1, ανάλογα με το αν έλαβε τελευταίο πακέτο από το LNK1 ή το LNK2. Άρα, όταν έρθει η απάντηση από το PC1, θα εγκλωβιστεί αενάως μεταξύ των B1, B2 αφού το PC3 θα φαίνεται να είναι είτε στο LNK1 είτε στο LNK2.

### **3.12**

Γίνεται συνέχεια η ερώτηση “who-has 192.168.1.1 tell PC3” και δίνεται η απάντηση “192.168.1.1 is-at 08:00:27:f0:ee:f2”.

### **3.13**

Λόγω του βρόχου που έχει δημιουργηθεί μεταξύ των B1, B2, το ARP Request, το οποίο και είναι broadcast, προωθείται συνέχεια εκτός του βρόχου, οπότε και λαμβάνεται από το PC2, η δε MAC προορισμού είναι η 08:00:27:6b:3b:db.

### **3.14**

Για τον ίδιο λόγο με επάνω, τα πακέτα βγαίνουν και εκτός βρόχου, οπότε γίνεται επανειλημμένα το ερώτημα και λαμβάνεται επίσης επανειλημμένα η απάντηση.

### **3.15**

Διότι πλέον η MAC του PC3 είναι είτε στην em1/em2 (LNK1/LNK2) για το B1, είτε στην em0/em1 (LNK1/LNK2) για το B2, άρα δε φεύγει ποτέ εκτός βρόχου. Φεύγουν μόνο τα πακέτα που είναι για broadcast.

## **Άσκηση 4: Δικτύωση Internal**

### **4.1**

Εκτελούμε “ifconfig bridge1 destroy”, “ifconfig bridge1 create” και “ifconfig bridge2 destroy”, “ifconfig bridge0 create” στα B1 και B2 αντίστοιχα.

### **4.2**

Ενεργοποιούμε τις κάρτες με τις διαδοχικές εντολές “ifconfig em0 up”, “ifconfig em1 up”, “ifconfig em2 up” στο B1. Εκτελούμε αμέσως μετά την εντολή “ifconfig lagg0 create”.

### **4.3**

Εκτελούμε στο B1 την εντολή “ifconfig lagg0 up laggport em1 laggport em2”.

### **4.4**

Όμοια στο B2.

#### 4.5

Εκτελούμε στο B1 την εντολή **"ifconfig bridge1 addm em0 addm lagg0 up"**.

#### 4.6

Αντίστοιχα στο B2 **"ifconfig bridge2 addm em2 addm lagg0 up"**.

#### 4.7

Κάνοντας ping από το PC2 στο PC3, βλέπουμε στην καταγραφή του PC1 το ARP Request στο οποίο το PC3 ρωτάει για την MAC του PC2. Αυτό που συνέβη είναι πως, δεδομένου ότι είχαμε καθαρίσει όλους τους ARP πίνακες, όταν πήγαμε να στείλουμε πακέτα από το PC2 στο PC3, δεδομένου ότι ο πρώτος δε γνώριζε τη MAC του 2<sup>ου</sup> έκανε broadcast στο LAN2 ένα ARP request για να τη μάθει. Από εκεί, το B2 την έκανε επίσης broadcast προς κάθε άλλη θύρα, εν προκειμένω προς το lagg0, από όπου και το παρέλαβε στο δικό του lagg0 το B1. Τέλος, το B1 το έκανε broadcast στο LAN1, από όπου και παρελήφθη από το PC1.

#### 4.8

Εκτελούμε **"tcpdump -i em0"** στο PC1.

#### 4.9

Το ping είναι επιτυχές και παρατηρούμε ARP request/reply όπως φαίνεται παρακάτω:

```
root@PC1:~ # tcpdump -i em0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on em0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:21:44.707642 ARP, Request who-has 192.168.1.1 tell PC3, length 46
19:21:44.707660 ARP, Reply 192.168.1.1 is-at 08:00:27:f0:ee:f2 (oui Unknown), length 28
19:21:44.708343 IP PC3 > 192.168.1.1: ICMP echo request, id 7172, seq 0, length 64
19:21:44.708351 IP 192.168.1.1 > PC3: ICMP echo reply, id 7172, seq 0, length 64
```

#### 4.10

Εκτελούμε **"tcpdump -i em1"** στο B1 και **"tcpdump -i em1"** στο B2. Παρατηρούμε πως τα πακέτα εμφανίζονται στο LNK1 (em1 του B1, em1 του B2). Το προεπιλεγμένο πρωτόκολλο συνάθροισης είναι το failover και μελετώντας την τεκμηρίωσή του, βλέπουμε πως η κίνηση μεταφέρεται μέσα από το master port, το οποίο εν προκειμένω είναι το em1 στο B1 και το em0 στο B2, καθώς αυτές ήταν οι πρώτες διεπαφές που προστέθηκαν στις συσκευές συνάθροισης. Παρακάτω, βλέπουμε τα Master Ports των B1 και B2 για να επαληθεύσουμε τα παραπάνω:

```

lagg0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, VLAN_HW
FILTER>
        ether 08:00:27:b9:ac:d6
        laggproto failover lagghash 12,13,14
        laggport: em1 flags=5<MASTER,ACTIVE>
        laggport: em2 flags=0<>
        groups: lagg
        media: Ethernet autoselect
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>

lagg0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, VLAN_HW
FILTER>
        ether 08:00:27:da:ee:00
        laggproto failover lagghash 12,13,14
        laggport: em0 flags=5<MASTER,ACTIVE>
        laggport: em1 flags=0<>
        groups: lagg
        media: Ethernet autoselect
        status: active
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>

```

### Τεκμηρίωση failover

<b>failover</b>	Sends traffic only through the active port. If the master port becomes unavailable, the next active port is used. The first interface added is the master port; any interfaces added after that are used as failover devices.
-----------------	---

#### 4.11

Ενώ έτρεχε το προηγούμενο ping και οι καταγραφές στα B1 και B2, αποσυνδέουμε το καλώδιο από την διεπαφή em1 του B1 και παρατηρούμε αμέσως πως τα πακέτα ICMP σταμάτησαν να καταγράφονται στο tcpdump του B1 και ξεκίνησαν να λαμβάνονται από το tcpdump του B2 (απενεργοποιούμε στη συνέχεια και την διεπαφή em0 του B2). Πλέον, η κίνηση καταγράφεται στο LNK2 (στο em1 του B2 που έτρεχε πριν), για τον ίδιο λόγο που αναφέραμε παραπάνω, το πρωτόκολλο της συνάθροισης που είναι ορισμένο στο failover, το οποίο και μας λέει πως όταν συμβεί κάποια βλάβη σε master port, γίνεται master port η διεπαφή που προστέθηκε αμέσως μετά αυτής που ήταν προηγουμένως master.

#### 4.12

Όπως αναμέναμε από το προκαθορισμένο failover πρωτόκολλο, με την επανασύνδεση της γραμμής LNK1, η κίνηση πλέον διοχετεύεται ξανά από εκεί.

## Άσκηση 5: Αποφυγή βρόγχων

### 5.1

Εκτελούμε τις εντολές “ifconfig bridge1/2 destroy”, “ifconfig lagg0 destroy”, “ifconfig em0 down”, “ifconfig em1 down”, “ifconfig em2 down” στα B1/B2 αντίστοιχα.

### 5.2

Εκτελούμε τις εντολές “ifconfig em0 up”, “ifconfig em1 up”, “ifconfig em2 up”, “ifconfig bridge1 create”, “ifconfig bridge1 addm em0 addm em1 addm em2 up” στο B1.

### 5.3

Αντίστοιχα στο B2.

### 5.4

Εκτελούμε “ifconfig bridge1 stp em0 stp em1 stp em2” στο B1.

### 5.5

Εκτελούμε “ifconfig bridge2 stp em0 stp em1 stp em2” στο B2.

### 5.6

Έχουμε:

- B1: priority → 32.768, id → 08:00:27:68:1e:90 ⇒  
**Bridge1\_ID = 32768.08:00:27:68:1e:90**
- B2: priority → 32.768, id → 08:00:27:55:8d:32 ⇒  
**Bridge2\_ID = 32768.08:00:27:55:8d:32**

```
bridge1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 02:9e:f6:6b:8b:01
        id 08:00:27:68:1e:90 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 08:00:27:55:8d:32 priority 32768 ifcost 20000 port 2
```

```
bridge2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 02:9e:f6:6b:8b:02
        id 08:00:27:55:8d:32 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 08:00:27:55:8d:32 priority 32768 ifcost 0 port 0
```

### 5.7

Αφού Bridge1\_ID > Bridge2\_ID γέφυρα ρίζα του επικαλύπτοντος δένδρου η B2.

## 5.8

Οι καταστάσεις και οι ρόλοι των διεπαφών της γέφυρας ρίζας περιγράφονται παρακάτω:

- **em0**: State → Forwarding, Role → Designated
- **em1**: State → Forwarding, Role → Designated
- **em2**: State → Forwarding, Role → Designated

Επομένως, όλες οι θύρες είναι πλήρως λειτουργικές και προωθούν προς τμήματα LAN.

```
member: em2 flags=1e7<LEARNING,DISCOVER,STP,EDGE,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 3 priority 128 path cost 20000 proto rstp
        role designated state forwarding
member: em1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 2 priority 128 path cost 20000 proto rstp
        role designated state forwarding
member: em0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 1 priority 128 path cost 20000 proto rstp
        role designated state forwarding
groups: bridge
nd6 options=9<PERFORMNUD,IFDISABLED>
```

## 5.9

Με την εντολή “**ifconfig bridge1**” στο B1 βλέπουμε πως ριζική θύρα είναι αυτή στο LNK1 (em1).

```
member: em2 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 3 priority 128 path cost 20000 proto rstp
        role alternate state discarding
member: em1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 2 priority 128 path cost 20000 proto rstp
        role root state forwarding
member: em0 flags=1e7<LEARNING,DISCOVER,STP,EDGE,AUTOEDGE,PTP,AUTOPTP>
        ifmaxaddr 0 port 1 priority 128 path cost 20000 proto rstp
        role designated state forwarding
groups: bridge
nd6 options=9<PERFORMNUD,IFDISABLED>
```

## 5.10

Η θύρα em2 του B1 έχει ρόλο alternate και state discarding, δηλαδή είναι εναλλακτική της ριζικής θύρας για τη διαδρομή προς τη γέφυρα ρίζα και δεν αποστέλλει πλαίσια.

## 5.11

Η θύρα em0 του B1 έχει ρόλο designated και state forwarding, συνεπώς είναι πλήρως λειτουργική και προωθεί προς τμήμα LAN (προς το LAN1 συγκεκριμένα).



### 5.12

Κάθε 2 δευτερόλεπτα. Παρακάτω η καταγραφή:

```
root@PC:~ # tcpdump -vvve -i em0
tcpdump: listening on em0, link-type EN10MB (Ethernet), capture size 262144 byte
s
20:45:59.748706 08:00:27:da:ee:00 (oui Unknown) > 01:80:c2:00:00:00 (oui Unknown
), 802.3, length 39: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, c
trl 0x03: STP 002.1w, Rapid STP, Flags [Learn, Forward], bridge-id 8000.08:00:27
:55:8d:32.8001, length 36
    message-age 0.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15
.00s
    root-id 8000.08:00:27:55:8d:32, root-pathcost 0, port-role Designated
20:46:01.747951 08:00:27:da:ee:00 (oui Unknown) > 01:80:c2:00:00:00 (oui Unknown
), 802.3, length 39: LLC, dsap STP (0x42) Individual, ssap STP (0x42) Command, c
trl 0x03: STP 002.1w, Rapid STP, Flags [Learn, Forward], bridge-id 8000.08:00:27
:55:8d:32.8001, length 36
    message-age 0.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15
.00s
    root-id 8000.08:00:27:55:8d:32, root-pathcost 0, port-role Designated
```

### 5.13

Παρατηρούμε πως είναι ενθυλακωμένα σε πλαίσια IEEE 802.3.

### 5.14

MAC<sub>SOURCE</sub> = 08:00:27:da:ee:00 και MAC<sub>DESTINATION</sub> = 01:80:c2:00:00:00.

### 5.15

Ανήκει στη διεπαφή em0 (LNK1).

### 5.16

Είναι multicast (01:80:c2:00:00:00), καθώς η σειρά με την οποία θα μεταδοθεί η διεύ-  
θυνση είναι πρώτα το LSB του πρώτου byte, άρα το 1.

### 5.17

Κατεγράφησαν οι εξής τιμές:

- Root ID: 8000.08:00:27:55:8d:32 (8000<sub>hex</sub> = 32768<sub>dec</sub>)
- Bridge ID: 8000.08:00:27:55:8d:32.8001
- Root Path Cost: 0

### 5.18

Από την Bridge ID (8000.08:00:27:55:8d:32.8001) προτεραιότητα είναι το πρώτο μέρος με τιμή 8000, το οποίο είναι σε δεκαεξαδική μορφή και μεταφράζεται σε 32.768 σε δε-  
καδική.

### **5.19**

Το δεύτερο μέρος είναι η MAC διεύθυνση της γέφυρας ρίζας και το τρίτο η id της θύρας από την οποία εκπέμπονται τα πλαίσια BPDUs με τιμή  $8001_{16} = 32769_{10}$ . Εκτελώντας `ifconfig bridge2` βλέπουμε πως η `em0` έχει port 1, επομένως το 8001 προκύπτει ως άθροισμα της προτεραιότητας με τον αριθμό της θύρας.

### **5.20**

Ναι, παρατηρούμε και από την άλλη γέφυρα.

### **5.21**

Κάνοντας καταγραφές στο B1, παρατηρούμε πως πηγές των BPDUs είναι οι `em0` (`tcpdump -i em1`) και `em1` (`tcpdump -i em2`) της B2, καθώς και η `em0` (port 1) του B1 που είναι και η ζητούμενη (`tcpdump -i em0`).

### **5.22**

Από την καταγραφή της `em0`, έχουμε τα εξής:

- Root ID: 8000.08:00:27:55:8d:32
- BridgeID: 8000.08:00:27:68:1e:90.8002
- Root Path Cost: 20.000

### **5.23**

Ναι, επιτυγχάνει κανονικά.

### **5.24**

Αποσυνδέουμε το LNK1 από το B1. Παρατηρούμε ότι πέρασαν συνολικά περίπου 6 δευτερόλεπτα. Η τιμή αυτή είναι αναμενόμενη και ίση με το 3πλάσιο του hello time, το οποίο είναι 2 δευτερόλεπτα.

### **5.25**

Αυτή τη φορά δε φαίνεται να υπάρχει διακοπή της επικοινωνίας.

## Άσκηση 6: Ένα πιο πολύπλοκο δίκτυο με εναλλακτικές διαδρομές

### B1

**em0:** LAN1

**em1:** LNK1

**em2:** LNK2

**em3:** LNK3

### B2

**em0:** LAN2

**em1:** LNK1

**em2:** LNK2

**em3:** LNK4

### B3

**em0:** LAN3

**em1:** LNK3

**em2:** LNK4

### 6.1

Κατασκευάζουμε εξ αρχής τις γέφυρες, επομένως, εκτελούμε στο B1 τις εντολές **“ifconfig em0 up”, “ifconfig em1 up”, “ifconfig em2 up”, “ifconfig em3 up”, “ifconfig bridge1 create”, “ifconfig bridge1 addm em0 addm em1 addm em2 addm em3 up”** και **“ifconfig bridge1 stp em0 stp em1 stp em2 stp em3”**.

### 6.2

Αντίστοιχα εκτελούμε στο B2 **“ifconfig em0 up”, “ifconfig em1 up”, “ifconfig em2 up”, “ifconfig em3 up”, “ifconfig bridge2 create”, “ifconfig bridge2 addm em0 addm em1 addm em2 addm em3 up”** και **“ifconfig bridge2 stp em0 stp em1 stp em2 stp em3”**.

### 6.3

Αντίστοιχα εκτελούμε στο B3 **“ifconfig em0 up”, “ifconfig em1 up”, “ifconfig em2 up”, “ifconfig bridge3 create”, “ifconfig bridge3 addm em0 addm em1 addm em2 up”** και **“ifconfig bridge3 stp em0 stp em1 stp em2”**.

### 6.4

Ναι, είναι επιτυχές.

### 6.5

Η γέφυρα bridge1 είναι ήδη ρίζα του δένδρου, ωστόσο, θα δίνουμε την εντολή **“ifconfig bridge1 priority 0”**.

### 6.6

Το path cost για τις ζεύξεις LNK1 (em1), LNK2 (em2) και LNK4(em3) στο B2 είναι ίδιο και ίσο με 20.000, το οποίο είναι κόστος για bandwidth 1Gbps, όπως και είναι οι κάρτες δικτύου μας.

## 6.7

Τα root path cost που λαμβάνει στα πλαίσια BPDU από τις γέφυρες 1 και 2 είναι αντίστοιχα 0 και 20.000 (τα διαβάζουμε με **"tcpdump -vei em1"** και **"tcpdump -vei em2"** αντίστοιχα). Η πρώτη τιμή είναι 0 επειδή η γέφυρα 1 είναι ρίζα, επομένως και δεν υπάρχει κόστος από αυτή μέχρι τη γέφυρα ρίζα, ενώ η δεύτερη είναι 20.000 επειδή το Bandwidth της κάρτας δικτύου της ριζικής θύρας em0 είναι 1000Mbit..

## 6.8

Εκτελώντας **"ifconfig bridge3"** βλέπουμε πως ριζική θύρα είναι η em1 (LNK3) και αυτό διότι από εκεί πάει άμεσα στο B1, επομένως έχει το μικρότερο δυνατό κόστος (δεδομένου ότι όλες οι διαδρομές έχουν κόστος 20.000).

## 6.9

Όσον αφορά τη θύρα στο LNK4 για το B3, ο ρόλος της είναι designated και η κατάστασή της forwarding (αν και θα αναμέναμε role alternate state discarding), ωστόσο στο B2 η θύρα στο LNK4 έχει ρόλο alternate και κατάσταση discarding.

## 6.10

Κάνοντας **"tcpdump -vvvei em0"** στο PC3 βλέπουμε πως καταγράφονται BPDU πλαίσια με root path cost ίσο με 20.000.

## 6.11

Κάνουμε **"ping PC3"** από το PC1.

## 6.12

Το κόστος μέσω της LNK4 από το B3 προς τη γέφυρα ρίζα αναμένεται πως είναι 20.000 (κόστος LNK4) + 20.000 (κόστος LNK1) = 40.000. Άρα θα θέσουμε μια τιμή μεγαλύτερη από 40.000 στο υπάρχον κόστος της (20.000). Εκτελούμε, επομένως, την εντολή **"ifconfig bridge3 ifpathcost em1 40001"**, και παρατηρούμε (**"ifconfig bridge3"**) πως ριζική πλέον θύρα στο B3 είναι η em2 (LNK4).

## 6.13

Πήρε περίπου 4 δευτερόλεπτα (διαφορά ώρας από του τελευταίου πακέτου πριν την αλλαγή και του αμέσως επόμενου με **"tcdump icmp -vei"** em0 στο PC3, αφαιρώντας ένα δευτερόλεπτο για το interval των πακέτων icmp που στέλνει το ping).

## 6.14

Πλέον ο ρόλος της em1 (LINK3) στη B3 είναι alternate και το state της discarding, που σημαίνει πως είναι μια εναλλακτική της ριζικής θύρας για τη διαδρομή προς τη γέφυρα

ρίζα, αλλά προς το παρόν δεν αποστέλλει πλαίσια χρηστών. Στο B2 η em3 (LNK4) έχει ρόλο designated και κατάσταση forwarding.

#### **6.15**

Δεν παρατηρείται κάποια διαφορά στις παραμέτρους που λαμβάνει η bridge3 (“tcpdump -vei em1” και “tcpdump -vei em2”).

#### **6.16**

Όσον αφορά τα παραγόμενα BPDU από τη γέφυρα bridge3, πλέον το root path cost έχει τιμή 40.000 (“tcpdump -vei em0”).

#### **6.17**

Χρειάστηκαν περίπου 7-8 δευτερόλεπτα (όμοια με 6.13).

#### **6.18**

Πέρασαν περίπου 23 με 24 δευτερόλεπτα.

#### **6.19**

Η αρχική θύρα που είχαμε (em0, port1) έχει ρόλο designated και κατάσταση forwarding, ενώ η em3, port 3 που προσθήσαμε τώρα έχει ρόλο disabled και κατάσταση discarding.

#### **6.20**

Διαλέξαμε την αρχική της τιμή αφού στο LINK3 βρίσκεται και η B1 οπότε θα επιλεγεί ως ριζική, καθώς το κόστος μέσω της LNK4 από το B3 προς τη γέφυρα ρίζα είναι 40000, ενώ μέσω της LINK3 είναι πλέον 20000.

## **Άσκηση 7: Εικονικά τοπικά δίκτυα VLAN**

Ορίζουμε τις διεπαφές των γεφυρών ως εξής:

### **B1**

em0: LAN1

em1: LNK1

em2: LNK2

em3: LNK3

### **B2**

em0: LAN2

em1: LNK1

em2: LNK2

em3: LNK4

### **B3**

em0: LAN3

em1: LNK3

em2: LNK4

### 7.1

Εκτελούμε τις εντολές **“ifconfig em0.5 create inet 192.168.5.1/24”** και **“ifconfig em0.6 create inet 192.168.6.1/24”** στο PC1.

### 7.2

Εκτελούμε στο B1 τις εντολές **“ifconfig em0.5 create”** και **“ifconfig em0.6 create”**.

### 7.3

Εκτελούμε στο B1 τις εντολές **“ifconfig em1.6 create”** και **“ifconfig em3.5 create”**.

### 7.4

Εκτελούμε στο PC2 την εντολή **“ifconfig em0.6 create inet 192.168.6.2/24”**.

### 7.5

Εκτελούμε στο B2 τις εντολές **“ifconfig bridge2 deletem em2”**, **“ifconfig em0.6 create”** και **“ifconfig em1.6 create”**.

### 7.6

Εκτελούμε στο PC3 την εντολή **“ifconfig em0.5 create inet 192.168.5.3/24”**.

### 7.7

Εκτελούμε στο B3 τις εντολές **“ifconfig em0.5 create”** και **“ifconfig em1.5 create”**.

### 7.8

Ναι, μπορούμε κανονικά.

### 7.9

Εκτελούμε στο B1 την εντολή **“ifconfig bridge1 –stp em0”**.

### 7.10

Εκτελούμε στο PC1 την εντολή **“tcpdump –vvvxi em0”**.

### 7.11

Ethertype: ARP (0x0806) και Ethertype: IPv4 (0x0800).

```
01:22:12.568782 08:00:27:e4:eb:5a (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.1 tell PC2, length 46
    0x0000: 0001 0800 0604 0001 0800 27e4 eb5a c0a8
    0x0010: 0102 0000 0000 0000 c0a8 0101 0000 0000
    0x0020: 0000 0000 0000 0000 0000 0000 0000
01:22:12.568797 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:e4:eb:5a (oui Unknown), ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.1 is-at 08:00:27:f0:ee:f2 (oui Unknown), length 28
    0x0000: 0001 0800 0604 0002 0800 27f0 eef2 c0a8
    0x0010: 0101 0800 27e4 eb5a c0a8 0102
01:22:12.569314 08:00:27:e4:eb:5a (oui Unknown) > 08:00:27:f0:ee:f2 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 13612, offset 0, flags [none], proto ICMP (1), length 84)
    PC2 > 192.168.1.1: ICMP echo request, id 5124, seq 0, length 64
    0x0000: 4500 0054 352c 0000 4001 c229 c0a8 0102
    0x0010: c0a8 0101 0800 41d9 1404 0000 0000 2dee
    0x0020: 20c1 6070 0809 0a0b 0c0d 0e0f 1011 1213
    0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
    0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
    0x0050: 3435 3637
```

### 7.12

Πλέον, πριν των παραπάνω ethertypes καταγράφεται το ethertype 802.1Q, το οποίο και είναι η VLAN ετικέτα.

```
01:38:04.238273 08:00:27:e4:eb:5a (oui Unknown) > Broadcast, ethertype 802.1Q (0x8100), length 64: vlan 6, p 0, ethertype ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.6.1 tell 192.168.6.2, length 46
    0x0000: 0001 0800 0604 0001 0800 27e4 eb5a c0a8
    0x0010: 0602 0000 0000 0000 c0a8 0601 0000 0000
    0x0020: 0000 0000 0000 0000 0000 0000 0000
01:38:04.238291 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:e4:eb:5a (oui Unknown), ethertype 802.1Q (0x8100), length 46: vlan 6, p 0, ethertype ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.6.1 is-at 08:00:27:f0:ee:f2 (oui Unknown), length 28
    0x0000: 0001 0800 0604 0002 0800 27f0 eef2 c0a8
    0x0010: 0601 0800 27e4 eb5a c0a8 0602
01:38:04.238746 08:00:27:e4:eb:5a (oui Unknown) > 08:00:27:f0:ee:f2 (oui Unknown), ethertype 802.1Q (0x8100), length 102: vlan 6, p 0, ethertype IPv4, (tos 0x0, ttl 64, id 13613, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.6.2 > 192.168.6.1: ICMP echo request, id 10500, seq 0, length 64
    0x0000: 4500 0054 352d 0000 4001 b828 c0a8 0602
    0x0010: c0a8 0601 0800 ed6c 2904 0000 0000 31a6
    0x0020: 1456 b08f 0809 0a0b 0c0d 0e0f 1011 1213
    0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
    0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
    0x0050: 3435 3637
```

### 7.13

Πλέον η τιμή του ethertype είναι 0x8100, ωστόσο παρατηρούμε πως είναι ενθυλακωμένος και ο τύπος που είχαμε δει πιο πριν (ethertype ARP) και (ethertype IPv4) αμέσως μετά.

### 7.14

Στο πεδίο μεταξύ της MAC προορισμού και του ethertype που είχαμε χωρίς το VLAN.

### 7.15

Εκτελούμε “tcpdump -vvvex -i em0.5” στο PC1.

### 7.16

Παρατηρούμε πως τα ethertypes έχουν ξανά τις τιμές 0x0806 (ARP) και 0x0800 (IPv4) στα πλαίσια που μεταφέρουν τα ARP και τα ICMP αντίστοιχα.

```
01:48:58.829926 08:00:27:6b:3b:db (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.5.1 tell 192.168.5.3, length 46
    0x0000: 0001 0800 0604 0001 0800 276b 3bdb c0a8
    0x0010: 0503 0000 0000 0000 c0a8 0501 0000 0000
    0x0020: 0000 0000 0000 0000 0000 0000 0000
01:48:58.829940 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:6b:3b:db (oui Unknown), ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Reply 192.168.5.1 is-at 08:00:27:f0:ee:f2 (oui Unknown), length 28
    0x0000: 0001 0800 0604 0002 0800 27f0 eef2 c0a8
    0x0010: 0501 0800 276b 3bdb c0a8 0503
01:48:58.830580 08:00:27:6b:3b:db (oui Unknown) > 08:00:27:f0:ee:f2 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 38198, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.5.3 > 192.168.5.1: ICMP echo request, id 12548, seq 0, length 64
    0x0000: 4500 0054 9536 0000 4001 5a1e c0a8 0503
    0x0010: c0a8 0501 0800 c9fe 3104 0000 0000 342f
    0x0020: 3ab7 a313 0809 0a0b 0c0d 0e0f 1011 1213
    0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
    0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
    0x0050: 3435 3637
01:48:58.830586 08:00:27:f0:ee:f2 (oui Unknown) > 08:00:27:6b:3b:db (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 36452, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.5.1 > 192.168.5.3: ICMP echo reply, id 12548, seq 0, length 64
```

### 7.17

Εκτελούμε “ifconfig bridge1 stp em0” στο B1 και “tcpdump -vvvex -i em0” στο PC1.

### 7.18

Είναι διαφορετικού τύπου, καθώς πριν είχαμε EtherType, επομένως Ethernet II πακέτα, ενώ πλέον στη θέση του πεδίου EtherType έχουμε Length.

### 7.19

Θα χρησιμοποιούσαμε το φίλτρο “not stp”.