Ονοματεπώνυμο: Χαράλαμπος Καμπουγέρης **Όνομα PC/ΛΣ:** DESKTOP-N90CRE0

Ομάδα: 1, Τρίτη 10:45-13:30, Αιθ.Α4 **Ημερομηνία:** 13/03/2024

Εργαστηριακή Άσκηση 3
Τοπικά δίκτυα και μεταγωγείς LAN

Άσκηση 1

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 Εκτελώντας τα κατάλληλα ping (ping 192.168.1.2 από το PC1 στο PC2 και ping 192.168.1.1 από το PC2 στο PC1), παρατηρούμε και από τα δύο μηχανήματα πως τα πακέτα που έστειλαν 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.

1.5 Εκτελούμε στο μηχάνημα B1: "ifconfig bridge0 create" για να δημιουργήσουμε τη γέφυρα και στη συνέχεια προσθέτουμε τις διεπαφές em0 και em1 με την εντολή "ifconfig bridge0 addm em0 addm em1 up"

1.6 Ναι αυτή τη φορά επικοινωνούν

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 και δεν προσθέτει βήματα

1.8 Ο πίνακας ARP του αντίστοιχου VM περιέχει τη δική του εγγραφή και την διεύθυνση MAC του άλλου VM (λείπει η διεύθυνση MAC της γέφυρας)

Για το PC1:

```
root@PC:~ # arp -a
? (192.168.1.1) at 08:00:27:43:dc:e5 on em0 permanent [ethernet]
? (192.168.1.2) at 08:00:27:d9:4b:e0 on em0 expires in 870 seconds [ethernet]
```

Για το PC2:

```
root@PC:~ # arp -a
? (192.168.1.1) at 08:00:27:43:dc:e5 on em0 expires in 649 seconds [ethernet]
? (192.168.1.2) at 08:00:27:d9:4b:e0 on em0 permanent [ethernet]
```

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 (μεγαλύτερους) χρόνους σύλληψης):

```
root@PC:~ # tcpdump -i em0 -ev
tcpdump: listening on em0, link-type EN10MB (Ethernet), capture size 262144 byte
s
00:44:23.412109 08:00:27:43:dc:e5 (oui Unknown) > 08:00:27:d9:4b:e0 (oui Unknown
), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 24674, offset 0, fla
gs [none], proto ICMP (1), length 84)
    192.168.1.1 > 192.168.1.2: ICMP echo request, id 36355, seq 0, length 64
00:44:23.412767 08:00:27:d9:4b:e0 (oui Unknown) > 08:00:27:43:dc:e5 (oui Unknown
), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 46016, offset 0, fla
gs [none], proto ICMP (1), length 84)
    192.168.1.2 > 192.168.1.1: ICMP echo reply, id 36355, seq 0, length 64
```

Αυτό που παρατηρούμε, είναι πως το request πακέτο φεύγει από την MAC 08:00:27:d7:4b:e0 (PC1) προς την 08:00:27:43:dc:e5 (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:

```
root@PC:~ # traceroute 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 64 hops max, 40 byte packets
 1  192.168.1.2 (192.168.1.2)  1.316 ms  1.119 ms  1.230 ms
```

Αυτό συμβαίνει επειδή μια γέφυρα λειτουργεί στο επίπεδο σύνδεσης δεδομένων (Επίπεδο 2) του μοντέλου OSI και δεν περιλαμβάνει διευθύνσεις IP ή δρομολόγηση. Όπως έχουμε πει, για το δίκτυο, η γέφυρα είναι “διαφανής”

1.13 Εκτελούμε στο B1 την εντολή “tcpdump -i em1 -vvve ” και κάνουμε “ping -i 192.168.1.2” από το PC1.

1.14 ifconfig em0 192.168.2.1 → στο PC2.
Ναι, η γέφυρα τα προωθεί στη θύρα που ήταν πριν το PC2 καθώς υπάρχει η καταγραφή στο πίνακα προώθησης. Ωστόσο αν διαγράψουμε το arp table από το PC1 (arp -d -a) σταματάει να προωθεί ICMP πακέτα στο PC2

1.15 Όχι, το ping δεν είναι επιτυχές, γιατί πλέον δεν υπάρχει μηχάνημα με IP 192.168.1.2 το οποίο μπορεί να απαντήσει

1.16 Όχι, δεν μπορώ, γιατί η διεπαφή em2 δεν έχει προστεθεί στο bridge0

1.17 Εκτελούμε “ifconfig em2 up” και μετά “ifconfig bridge0 addm em2” στο VM B1

1.18 Τώρα 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.

1.20 Αυτή τη φορά, καταγράφηκε το εξής ARP request.

Ο λόγος που έγινε αυτό, είναι πως αφού το PC1 δεν ήξερε την MAC διεύθυνση που αντιστοιχεί στην IPPC3, 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.

1.21 Με την εντολή “ifconfig bridge0”.

1.22 ifconfig bridge0 addr

1.23 Αντιστοιχούν στα μηχανήματα PC1,PC2,PC3

1.24 ifconfig bridge0 flush

1.25 ifconfig bridge0 deletem em2

1.26 ifconfig bridge0 destroy

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

Άσκηση 2

2.1 ifconfig em0 192.168.1.x/24 , x = {1,2,3,4}

2.2 ifconfig bridge1 create , ifconfig em0 up , ifconfig em1 up , ifconfig bridge1 addm em0 addm em1 up

2.3 ifconfig bridge2 create , ifconfig em0 up , ifconfig em1 up , ifconfig bridge2 addm em0 addm em1 up

2.4 ifconfig bridge3 create , ifconfig em0 up , ifconfig em1 up , 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 -a -d”.

- PC1: 08:00:27:43:dc:e5

• PC2: 08:00:27:d9:4b:e0
• PC3: 08:00:27:aa:12:da
• PC4: 08:00:27:6f:73:3c

2.6 ifconfig bridgeX flush → X = {1, 2, 3}

2.7 tcpdump → για κάθε PC

2.8 Οι πίνακες προώθησης για το B1, B2, B3:

```
root@PC:~ # ifconfig bridge1 addr
08:00:27:d9:4b:e0 Vlan1 em1 546 flags=0<>
08:00:27:43:dc:e5 Vlan1 em0 546 flags=0<>
```

```
root@PC:~ # ifconfig bridge2 addr
08:00:27:d9:4b:e0 Vlan1 em0 540 flags=0<>
08:00:27:43:dc:e5 Vlan1 em0 540 flags=0<>
```

```
root@PC:~ # ifconfig bridge3 addr
08:00:27:43:dc:e5 Vlan1 em0 533 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 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 η B3, η οποία το καταχώρησε στο em0 (LNK2)
• Στο μεταξύ, το PC2 εξέπεμψε την απάντηση του στο LNK1, την οποία και έλαβε η B1 και η B2, καταχωρώντας έτσι την MAC_{PC2} η B1 στο em1 (LNK1) και η 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 και τη έχει καταχωρήσει στο arp table του. Επομένως, δε κάνει ARP request, με αποτέλεσμα το B3 να συνεχίζει να γνωρίζει μόνο την MAC του PC1. Οι πίνακες των B1, B2 δεν αλλάζουν επίσης καθόλου, καθώς ήδη γνώριζαν για τα PC1 και PC2.

2.11 Μετά από αυτό το ping όλες οι γέφυρες έχουν την MAC του PC4. 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 του PC4 να υπάρχει και στο B1.

2.12 Μετά από αυτό το ping όλες οι γέφυρες έχουν την MAC του PC3. Το PC3 κάνει ARP request γιατί δεν γνωρίζει την MAC του PC2. Η MAC του PC3 αποθηκεύεται στα B2,B3. Μέσω του ARP request το οποίο περνάει και από τη γέφυρα B2 προς το LNK1 γνωστοποιείται η MAC του και στο B1.

2.13 Εκτελούμε "ping 192.168.1.2" από τα PC1 και PC4

2.14 Το ping από το PC4 στο PC2 συνεχίζει κανονικά, καθώς το PC2 θεωρείται πλέον συνδεδεμένο με το PC4.

2.15 Το ping από το PC1 στο PC2 συνεχίζει να στέλνει ICMP Echo requests χωρίς όμως να λαμβάνει απάντηση. Αυτό συμβαίνει καθώς η γέφυρα B1 στέλνει συνεχώς τα πακέτα στο LNK1, εφόσον έχει αποθηκευμένη την παλιά θέση του PC2, ενώ επίσης το B2 δεν προωθεί τα πακέτα στο LNK2 καθώς και για αυτό βρίσκεται στο PC2 στο LNK1.

2.16 Παρατηρούμε ότι μετά το ping του PC2 προς το PC3 , το PC1 λαμβάνει απάντηση ξανά από το PC2. Αυτό που συνέβη είναι ότι με την εντολή ping στάλθηκε πακέτο στο PC3 μέσω του B3, το οποίο μεταδόθηκε στο LNK2. Εκεί, το B2 ενημέρωσε τον πίνακα του για τη θέση του PC2, ότι βρίσκεται πλέον στη θύρα em1. Έτσι, το επόμενο πακέτο που έστειλε το PC1, έφτασε στο B1 στάλθηκε στο LNK1 και από εκει στο B2 όπου και προωθήθηκε στο LNK2 για να φτάσει τελικά στο PC2.

2.17 Μέσω της εντολής "ifconfig bridge1" στο B1, βλέπουμε πως ο μέγιστος χρόνος που διατηρείται μια εγγραφή στον πίνακα προώθησης είναι 1200 seconds. Οπότε μετά από 20 λεπτά θα μάθαινε μόνη της η γέφυρα την αλλαγή.

Άσκηση 3

3.1 ifconfig bridge0 create , ifconfig bridge0 addm em0 addm em1 up

3.2 ifconfig bridge1 create , ifconfig bridge1 addm em0 addm em1 up

3.3
PC1: 08:00:27:43:dc:e5
PC2: 08:00:27:d9:4b:e0
PC3: 08:00:27:aa:12:da
Άδειασμα με arp -d -a

3.4 Παρατηρούμε πως καταγράφεται ARP κίνηση, με το PC2 να ρωτάει για την MAC της $IP_{PC3.}$ Πιο συγκεκριμένα, καθώς το PC2 δε γνώριζε τη MAC του PC3 για να στείλει αμέσως πακέτο, έστειλε ένα ARP request για να τη μάθει. Επειδή το PC3 βρίσκεται στο ίδιο LAN με αυτό, έμαθε αμέσως την απάντηση και στάλθηκε ICMP Echo μήνυμα. Παράλληλα όμως

προωθήθηκε το ARP Request μέσω του B2 στο LNK1 και από εκεί στο B1, το οποίο και το προώθησε στο PC1

3.5 PC3: ping 192.168.1.1

3.6 ifconfig em2 up , ifconfig bridgeX addm em2

3.7

```
root@PC:~ # ifconfig bridge1 addr
08:00:27:43:dc:e5 Vlan1 em0 1192 flags=0<>
08:00:27:aa:12:da Vlan1 em1 1192 flags=0<>
08:00:27:d9:4b:e0 Vlan1 em1 955 flags=0<>
```

```
root@PC:~ # ifconfig bridge2 addr
08:00:27:43:dc:e5 Vlan1 em0 1184 flags=0<>
08:00:27:aa:12:da Vlan1 em1 1184 flags=0<>
08:00:27:d9:4b:e0 Vlan1 em1 951 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.

3.9 tcpdump

3.10 Το ping δεν επιτυγχάνει

3.11 Αποσυνδέουμε τα καλώδια των LNK2 από τα B1, B2. Το PC1 εμφανίζεται ξανά στην em0 του B2, αλλά το PC3 εμφανίζεται στο em2 του B2 αντί του em1.
Αναλυτικά: το PC3 στέλνει το ARP request στο LAN2, το οποίο παραλαμβάνει το PC1 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 PC2 είναι στο em1 ή στο em2, 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 192.168.1.3” και δίνεται η απάντηση “192.168.1.1 is-at 08:00:27:43:dc:e5 ”.

3.13 Λόγω του βρόχου που έχει δημιουργηθεί μεταξύ των B1, B2, το ARP Request, το οποίο και είναι broadcast (“who-has 192.168.1.1 tell 192.168.1.3”), προωθείται συνέχεια εκτός του βρόχου, οπότε και λαμβάνεται από το PC2.

3.14 Όπως εξηγήσαμε στο **3.11** λόγω συνεχούς flouding των γεφυρών, τα πακέτα βγαίνουν και εκτός βρόχου, οπότε γίνεται επανειλημμένα το ερώτημα και λαμβάνεται επίσης επανειλημμένα η απάντηση.

3.15 Γιατί στο πίνακα προώθησης του B2 εχει καταγραφεί (λόγω του κύκλου) ότι το PC3 βρίσκεται στη θύρα em2 → LNK2 και έτσι το reply δε φεύγει ποτέ εκτός βρόχου. Φεύγουν μόνο τα πακέτα που είναι για broadcast, δηλαδή τα request.

Άσκηση 4

4.1 ifconfig bridgeX destroy , ifconfig emX down , ifconfig bridgeX create

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 Εκτελούμε στο B1 την εντολή "ifconfig bridge2 addm em1 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, ο πρώτος δε γνώριζε τη MAC του 2ου και 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 tcpdump → PC1

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

```
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
01:09:55.515722 ARP, Request who-has 192.168.1.1 tell 192.168.1.3, length 46
01:09:55.515753 ARP, Reply 192.168.1.1 is-at 08:00:27:43:dc:e5 (oui Unknown), le
ngth 28
01:09:55.518517 IP 192.168.1.3 > 192.168.1.1: ICMP echo request, id 36870, seq 0
, length 64
01:09:55.518531 IP 192.168.1.1 > 192.168.1.3: ICMP echo reply, id 36870, seq 0,
length 64
```

4.10 Εκτελούμε "tcpdump -i em1" στο B1 και "tcpdump -i em2" στο B2. Παρατηρούμε πως τα πακέτα εμφανίζονται στο LNK1 (em1 του B1, em0 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.

4.11 Απενεργοποιούμε τις em1 του B1 και em0 του B2. Παρατηρούμε ότι η κίνηση μεταφέρθηκε από το LNK1 στο LNK2

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

Άσκηση 5

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 → 08:00:27:3b:09:e7 => BridgeID1 = 32768.08:00:27:3b:09:e7
• B2: priority → 32.768, id → 08:00:27:80:0c:d1 => BridgeID2 = 32.768.08:00:27:80:0c:d1

5.7 Αφού BridgeID1 < BridgeID2 γέφυρα ρίζα του επικαλύπτοντος δένδρου η B1.

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.

5.9 Με την εντολή “ifconfig bridge2” βλέπουμε πως ριζική θύρα είναι αυτή στο LNK1 (em0).

```
root@PC:~ # ifconfig bridge2
bridge2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 58:9c:fc:00:63:07
        id 08:00:27:80:0c:d1 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 08:00:27:3b:09:e7 priority 32768 ifcost 20000 port 1
        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 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 root state forwarding
```

5.10 Η θύρα em2 του B2 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 em1 του B2 έχει ρόλο designated και state forwarding, συνεπώς είναι πλήρως λειτουργική και προωθεί προς τμήμα LAN (προς το LAN2 συγκεκριμένα).

5.12 tcpdump -i em0 -e -vvv → εκπέμποται κάθε 2 δευτερόλεπτα (hello-time)

5.13 Χρησιμοποιείται ενθυλάκωση IEEE 802.3

5.14 MAC_{source} = 08:00:27:3b:09:e7 και $MAC_{destination}$ = 01:80:c2:00:00:00.

5.15 Ανήκει στη διεπαφή em1 (LNK1).

5.16 Είναι multicast (01:80:c2:00:00:00), καθώς η σειρά με την οποία θα διαβαστεί η διεύθυνση είναι πρώτα το LSB του πρώτου byte, άρα το 1.

5.17Κατεγράφησαν τα εξής:
- Root ID: 8000.08:00:27:3b:09:e7 (8000hex = 32.768dec)
- Bridge ID: 8000.08:00:27:3b:09:e7.8001
- Root Path Cost: 0

5.18 Έχουμε την εξής καταγραφή στο B1, στο LNK2:

```
root@PC:~ # tcpdump -vvv -e -i em2
tcpdump: listening on em2, link-type EN10MB (Ethernet), capture size 262144 byte
s
02:25:35.597197 08:00:27:70:66:bc (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 802.1w, Rapid STP, Flags [Learn, Forward], bridge-id 8000.08:00:27
:3b:09:e7.8003, length 36
        message-age 0.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15
.00s
        root-id 8000.08:00:27:3b:09:e7, root-pathcost 0, port-role Designated
```

Από την Bridge ID (8000.08:00:27:3b:09:e7.8003) προτεραιότητα είναι το πρώτο μέρος με τιμή 8000, το οποίο είναι σε δεκαεξαδική μορφή και μεταφράζεται σε 32.768 σε δεκαδική/

5.19 Το δεύτερο μέρος της Bridge ID ανήκει στο route id

5.20 Το ID της θύρας από την οποία εκπέμπονται τα πλαίσια BPDU είναι το τελευταίο μέρος με τιμή $8003_{16} = 32.771_{10}$.
Εκτελώντας ifconfig bridge1 βλέπουμε πως η em2 έχει port 3, επομένως το 8003 προκύπτει ως άθροισμα της προτεραιότητας με τον αριθμό της θύρας.

5.21 Ναι, παρατηρούμε και από την άλλη γέφυρα

5.22 Κάνοντας καταγραφές στο B2, παρατηρούμε πως πηγές των BPDU είναι οι em1 (tcpdump -i em0) και em2 (tcpdump -i em2) της B1, καθώς και η em1 (port 2) του B2 που είναι και η ζητούμενη (tcpdump -i em1).

5.23 Από την καταγραφή της em1, έχουμε τα εξής:
- Root ID: 8000.08:00:27:3b:09:e7
- BridgeID: 8000.08:00:27:ac:fc:77.8002
- Root Path Cost: 20.000

5.24 Ναι, είναι επιτυχές

5.25 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.

5.26 Όχι δεν υπάρχει διακοπή της επικοινωνίας.

Άσκηση 6

6.1 Κατασκευάζουμε εξαρχής 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 δίναμε την εντολή "ifconfig bridge1 priority 0".

6.6 Τα path costs είναι όλα 20000. Επομένως από το τύπο 20 Tbps / bandwidth προκύπτει σωστά για bandwith = 1Gbps

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 -vvv -i em0" και "tcpdump - vvv -i em1" αντίστοιχα). Η πρώτη τιμή είναι 0 επειδή η γέφυρα 1 είναι ρίζα με αποτέλεσμα να μην 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 1Gbps.

6.8 Εκτελώντας "ifconfig bridge3" βλέπουμε πως ριζική θύρα είναι η em0 (LNK3) και αυτό καθώς από εκεί πηγαίνει άμεσα στο B1, επομένως έχει το μικρότερο δυνατό κόστος (δεδομένου ότι όλες οι διαδρομές έχουν κόστος 20.000).

6.9 Όσον αφορά στη θύρα στο LNK4, ο ρόλος της είναι alternate και η κατάστασή της discarding, επομένως, είναι μια εναλλακτική της ριζικής θύρας για τη διαδρομή προς τη γέφυρα ρίζα, η οποία ωστόσο δεν αποστέλλει πλαίσια στην παρούσα φάση.

6.10 Root-path cost = 20000

6.11 ping 192.168.1.3 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 θέτουμε μια τιμή μεγαλύτερη από 40.000 στο υπάρχον κόστος της (20.000). Εκτελούμε, επομένως, την εντολή "ifconfig bridge3 ifpathcost em0 40.001", και παρατηρούμε ("ifconfig bridge3") πως ριζική πλέον θύρα στο B3 είναι η em1 (LNK4).

```
root@PC:~ # ifconfig bridge3
bridge3: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 58:9c:fc:10:7e:14
        id 08:00:27:76:d6:75 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 08:00:27:3b:09:e7 priority 32768 ifcost 40000 port 2
        member: em2 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 3 priority 128 path cost 20000 proto rstp
                role designated state learning
        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=5c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 1 priority 128 path cost 40001 proto rstp
                role alternate state discarding
```

6.13 Πήρε περίπου 5 δευτερόλεπτα.

6.14 Πλέον ο ρόλος της διεπαφής της bridge3 στο LNK3 είναι alternate και το state της discarding, που σημαίνει πως είναι μια εναλλακτική της της ριζικής θύρας για τη διαδρομή προς τη γέφυρα ρίζα, αλλά προς το παρόν δεν αποστέλλει πλαίσια χρηστών.
Αντίθετα ο ρόλος της διεπαφής της bridge2 στο LNK4 είναι designated και το state της είναι forwarding, συνεπώς είναι πλήρως λειτουργική και προωθεί προς τμήμα LNK4

6.15 Δεν παρατηρείται κάποια διαφορά στις παραμέτρους που λαμβάνει η bridge3 ("tcpdump -vvv -i em0" και "tcpdump -vvv -i em1").

6.16 Ναι, πλέον η bridge3 φαίνεται να έχει root-path cost ίσο με 40000

6.17 Χρειάστηκαν περίπου 7 δευτερόλεπτα (3 hello-time).

6.18 Η επικοινωνία αποκαταστάθηκε πρακτικά άμεσα (δεν φαίνεται κάποια διακοπή με το μάτι)

6.19

```
        member: em3 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 4 priority 128 path cost 20000 proto rstp
                role backup state discarding
        member: em2 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 3 priority 128 path cost 20000 proto rstp
                role designated state forwarding
```

Όπως βλέπουμε, η 3η διεπαφή δηλαδή η em2 είναι πλήρως λειτουργική και προωθεί προς το τμήμα LAN3. Η 4η διεπαφή είναι σε ρόλο backup και state discarding, που σημαίνει ότι αυτή τη στιγμή δεν αποστέλλει πλαίσια αλλά λειτουργεί σε περίπτωση βλάβης στη σύνδεση της em2.

6.20 Διαλέγουμε μια τιμή κόστους <= 40.000 Επιλέγω 35000 και έχουμε:

```
root@PC:~ # ifconfig bridge3
bridge3: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 58:9c:fc:10:7e:14
        id 08:00:27:57:f1:bb priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 08:00:27:3b:09:e7 priority 32768 ifcost 35000 port 1
        member: em3 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 4 priority 128 path cost 20000 proto rstp
                role disabled state discarding
        member: em2 flags=1c7<LEARNING,DISCOVER,STP,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 alternate state discarding
        member: em0 flags=5c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                ifmaxaddr 0 port 1 priority 128 path cost 35000 proto rstp
                role root state forwarding
```

Άσκηση 7

7.1 Για το VLAN 5: ifconfig em0.5 create vlan 5 vlandev em0 inet 192.168.5.1/24
Για το VLAN 6: ifconfig em0.6 create vlan 5 vlandev em0 inet 192.168.6.1/24

7.2 Για το VLAN 5: ifconfig em0.5 create vlan 5
Για το VLAN 6: ifconfig em0.6 create vlan 5

7.3 ifconfig em1.6 create vlan 6
Ifconfig em3.5 create vlan 5

7.4 Εκτελούμε στο PC2 την εντολή "ifconfig em0.6 create vlan 6 vlandev em0 inet 192.168.6.2/24"

7.5 ifconfig em0.6 create vlan 6
ifconfig em1.6 create vlan 6

7.6 Εκτελούμε στο PC3 την εντολή "ifconfig em0.5 create vlan 5 vlandev em0 inet 192.168.5.3/24".

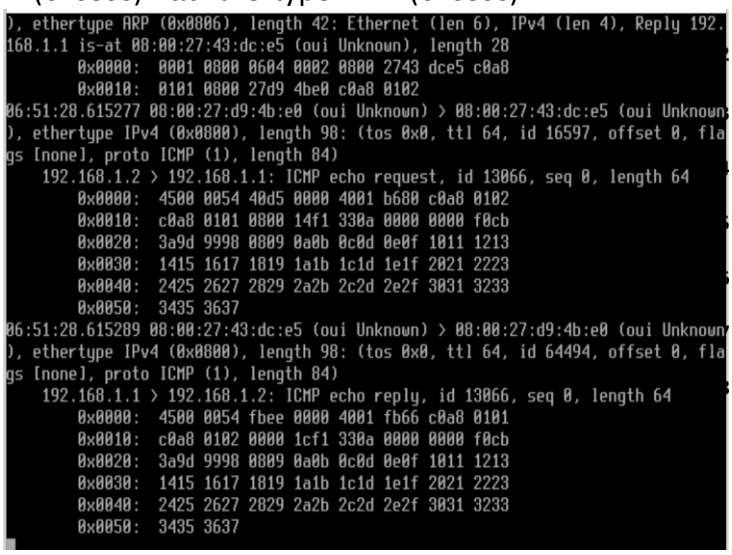
7.7 ifconfig em0.5 create vlan 5
ifconfig em2.5 create vlan 5

7.8 Ναι, μπορούμε

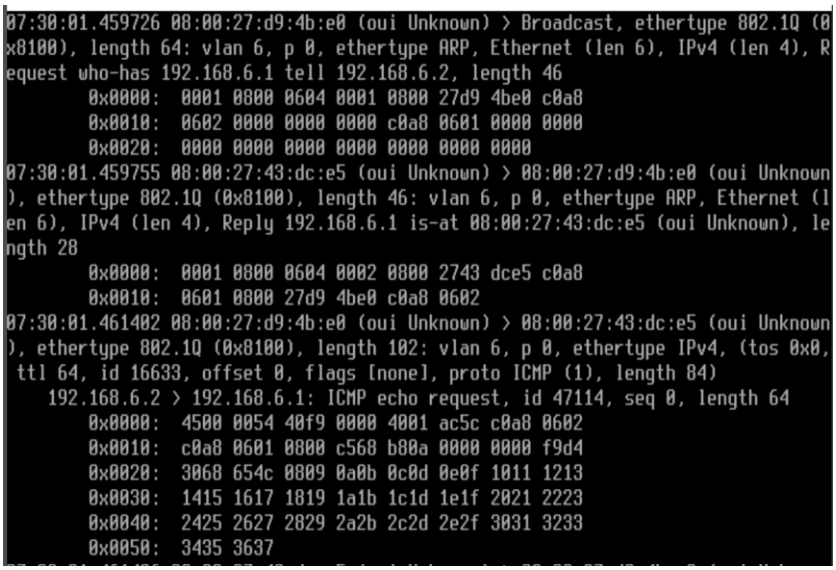
7.9 Εκτελούμε στο PC1 "ifconfig bridge1 -stp em0"

7.10 tcpdump -i em0 -e -vvv -x

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



7.12 Εκτός από τα παραπάνω ethertypes καταγράφεται το ethertype 802.1Q, το οποίο και είναι η VLAN ετικέτα, με αποτέσμα να έχουν μεγαλύτερο μέγεθος γιατί περιέχουν και το vlan tag (4 bytes)



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 4 bytes μετά το πεδίο length

7.15 tcpdump -i em0.5 -e -vvv -x

7.16 Ethertype ARP → 0806
Ethertype IPv4 → 0800
Όχι, δεν υπάρχει κάποιο σχετικό πεδίο

7.17 Εκτελούμε “ifconfig bridge1 stp em0” στο B1 και “tcpdump -i em0 -e -vvv -x ” στο PC1.

7.18 Στη θέση του Ethertype υπάρχει το 802.3. Υποδεικνύει ότι τα πλαίσια χρησιμοποιούν την original μορφή πλαισίωσης Ethernet IEEE 802.3.

7.19 Θα χρησιμοποιούσαμε το φίλτρο “not stp”