Ονοματεπώνυμο: Ανδρέας Στάμος (03120***)	**Όνομα PC:** linux / Ubuntu 22.04.2 LTS
Ομάδα: 1	**Ημερομηνία:** 27/02/2024

Εργαστηριακή Άσκηση 3
Τοπικά δίκτυα και μεταγωγείς LAN

Απαντήστε στα ερωτήματα στον χώρο που σας δίνεται παρακάτω και στην πίσω σελίδα εάν δεν επαρκεί. Το φυλλάδιο αυτό θα παραδοθεί στον επιβλέποντα.

Άσκηση 1

1.1 PC1: `ifconfig em0 192.168.1.1/24 up`

PC2: `ifconfig em0 192.168.1.2/24 up`

1.2 Με `ifconfig` βλέπουμε ότι οι `em2`, `em3` είναι σε `status: no carrier`. Συνεπώς ενεργοποιούμε τις `em0`, `em1` τρέχοντας:

```
ifconfig em0 up
ifconfig em1 up
```

1.3 Αποτυγχάνει. Τα PC1, PC2 βρίσκονται σε διαφοερτικά LAN και η γέφυρα ακόμα δεν έχει ρυθμιστεί ως γέφυρα.

1.4 Μέσω των MAC διευθύνσεων που μας δίνει το VirtualBox και αυτών που μας δίνει το `ifconfig` στην B1 βλέπουμε ότι η διεπαφή `em0` αντιστοιχεί στο LAN1 και η διεπαφή `em1` αντιστοιχεί στο LAN2.

Καταγράφουμε την κίνηση στο LAN1 με `tcpdump -n -i em0` στην B1. Εκτελούμε ping PC1 → PC2 και βλέπουμε ότι καταγράφαται πλαίσιο ARP Request για την MAC διεύθυνση που αντιστοιχεί στην IPv4 διεύθυνση 192.168.1.2. Δεν καταγράφαται ICMP κίνηση.

Όμοια (αν και συμμετρικά ισχύουν τα ίδια), καταγράφουμε την κίνηση στο LAN2 με `tcpdump -i em1` στην B1. Εκτελούμε ping PC2 → PC2 και βλέπουμε ότι καταγράφαται πλαίσιο ARP Request για την MAC διεύθυνση που αντιστοιχεί στην IPv4 διεύθυνση 192.168.1.2. Δεν καταγράφαται ICMP κίνηση.

Ο λόγος που δεν καταγράφαται ICMP κίνηση είναι πως τα PC1, PC2 βλέπουν στους πίνακες δρομολόγησης πως για να στείλουν στις διευθύνσεις 192.168.1.1, 192.168.1.2 αντίστοιχα πρέπει να στείλουν απευθείας στο υπόδικτυο της διεπαφής `em0`. Έτσι στέλνουν ARP Request για να βρουν την MAC διεύθυνση που αντιστοιχεί στην IPv4 διεύθυνση του παραλήπτη. Το ARP μήνυμα δεν το απαντά κανένα. Έτσι μετά από λίγο, οι αντίστοιχοι hosts θεωρούνται down, οπότε δεν στέλνεται φυσικά και μήνυμα ICMP.

1.5 Στην B1:

```
ifconfig bridge create
ifconfig bridge0 addm em0
ifconfig bridge0 addm em1
ifconfig bridge0 uo
```

1.6 Ναι. Τα pings PC1 → PC2 και PC2 → PC1 επιτυγχάνουν.

1.7 0. Το TTL μειώνεται όταν μεσολαβεί δρομολογητής, δηλαδή στο στρώμα δικτύου. Αντίθετα εμείς χρησιμοποιούμε γέφυρα, δηλαδή μεταγωγέα στο στρώμα ζεύξης δεδομένων, γεγονός που είναι αόρατο ως προς το στρώμα δικτύου.

1.8 Οι πίνακες είναι ίδιοι. Είναι λογικό, αφού βρίσκονται στο ίδιο υποδίκτυο, οπότε για να επικοινωνήσουν πρέπει να ξέρουν την ίδια αντιστοιχία IPv4 - MAC.

1.9 Θα επιχειρήσουμε να καταγραψούμε ταυτόχρονα την κίνηση. Θα χρησιμοποιήσουμε το **tmux** για να "σπάσουμε" την οθόνη σε 2 τερματικά. Ακολοθούμε την εξής διαδικασία στην B1:

1. Προσωρινά συνδεόμαστε στο διαδίκτυο (με NAT στην 3η διεπαφή) και με `pkg install tmux` εγκαθιστούμε το `tmux`.
2. Αποσυνδέουμε την 3η διεπαφή από το NAT και διαγράφουμε κάθε σχετική ρύθμιση στο FreeBSD.
3. Τρέχουμε `tmux`.

4. Σπάμε την οθόνη με `tmux split-window`. Από το πάνω στο κάτω pane μετακινούμαστε με `<CTRL> b <UPARROW>` (και αντίστροφα με `<DOWNARROW>`). Για να σκρολλάρουμε μέσα σε ένα pane μπαίνουμε σε copy mode με `<CTRL> b [` και έπειτα χρησιμοποιούμε τα βελάκια.
5. Τρέχουμε στο πάνω pane: `tcpdump -n -e -i em0`
6. Τρέχουμε στο κάτω pane: `tcpdump -n -e -i em1`

Πράγματι, παρατηρούμε ότι οι δύο καταγράφες είναι ίδιες, εκτός του χρόνου που εισάγεται μια (μικρή) καθυστέρηση για την μεταγωγή. Άρα πράγματι η γέφυρα μετάγει τα πλαίσια.

1.10 Όχι.

1.11 Όχι.

1.12 Όχι. Το `traceroute` ανιχνεύει δρομολογητές που μειώνουν το TTL, διαδικασία που συμβαίνει κατά την δρομολόγηση στο στρώμα δικτύου, ενώ εδώ έχουμε γέφυρα που μετάγει στο στρώμα ζεύξης δεδομένων.

1.13 Στο PC1: `ping 192.168.1.2`

Στην B1: `tcpdump -n -e -i em1`

1.14 Στο PC2: `ifconfig 192.168.2.1/24`

Παρατηρούμε στην καταγράφη στην γέφυρα B1 πως μηνύματα ICMP Echo Request εκπέμπονται από το PC1 στο LAN1 και μετάγονται από την γέφυρα στο LAN2. (Επίσης με αντίστοιχη καταγραφή στο PC2 βλέπουμε οτι τα μηνύματα φθάνουν στο PC2.)

1.15 Μετά την αλλαγή της διεύθυνσης του PC2, παρατηρούμε πως το `ping` στο PC1 σταματά να δείχνει replies, όμως δεν δείχει και `Host is down`. Ο λόγος που μηνύματα ICMP Echo Request μεταδίδονται είναι πως το PC1 έχει καταγράψει στον ARP πίνακά του μια αντιστοιχία για την διεύθυνση 192.168.1.2. Έτσι χωρίς να γνωρίζει ότι κάτι άλλαξε, συνεχίζει να στέλνει προς την MAC διεύθυνση αυτή. Αντίθετα, το PC2 παραλαμβάνει από το LAN2 ένα IPv4 πακέτο που δεν έχει για διεύθυνση παραλήπτη την IPv4 διεύθυνσή του, οπότε το αγνοεί, και συνεπώς, δεν στέλνει σχετικό ICMP Echo Reply.

1.16 Όχι, λαμβάνουμε `Host is down`, καθώς η γέφυρα B1 ακόμα δεν προωθεί πλαίσια στο LAN3.

1.17 Στην B1:

```
ifconfig em2 up
ifconfig bridge0 addm em2
```

1.18 Ναι.

1.19 Στην B1: `tcpdump -i em1 -e -n`

Παρατηρούμε ότι δεν εμφανίζονται ICMP μηνύματα στο LAN2. Αυτό που συμβαίνει είναι ότι μόλις το PC3 έστειλε το πρώτο ICMP Echo Reply, η γέφυρα B1 έμαθε ότι είναι στο LAN3, οπότε τα πλαίσια που το αφορούν τα στέλνει μόνο στο LAN3. (το γεγονός ότι το PC1 είναι στο LAN1 το ήξερε από πριν, αλλά ούτως ή άλλως και εδώ το μαθαίνει μόλις στείλει το πρώτο ICMP Echo Request)

1.20 Στα PC1, PC3: `arp -a -d`

Παρατηρούμε ότι εκτελώντας ping PC1 → PC3, στην καταγραφή στην B1 για το LAN2, εκπέμπεται ένα ARP Request. Αυτό που συμβαίνει είναι ότι το PC1 εκπέμπει ένα ARP Request για να μάθει την MAC διεύθυνση της IPv4 διεύθυνσης 192.168.1.3. Το ARP πλαίσιο έχει διεύθυνση broadcast FF:FF:FF:FF:FF:FF οπότε η γέφυρα το προωθεί παντού.

1.21 Στην B1: `ifconfig bridge0 | grep member`

1.22 Στην B1: `ifconfig bridge0 addr`

1.23 Στα PC1, PC2, PC3.

1.24 Στην B1: `ifconfig bridge0 flush`

1.25 Στην B1: `ifconfig bridge0 deletem em2`

1.26 Στην B1: `ifconfig bridge0 destroy`

1.27 Στα PC1, PC2, PC3: `ifconfig em0 delete`

Άσκηση 2

2.1 Στα PC1, PC2, PC3, PC4: `ifconfig em0 192.168.1.X/24 UP` όπου $X = 1, 2, 3, 4$ αντίστοιχα.

2.2 Στην B1:

```
ifconfig bridge1 create
ifconfig bridge1 addm em0
ifconfig bridge1 addm em1
ifconfig bridge1 up
```

2.3 Στην B2:

```
ifconfig bridge2 create
ifconfig bridge1 addm em0
ifconfig bridge1 addm em1
ifconfig bridge2 up
```

2.4 Στην B3:

```
ifconfig bridge3 create
ifconfig bridge1 addm em0
ifconfig bridge1 addm em1
ifconfig bridge3 up
```

2.5 Αυτοματοποιημένα στον host με την εξής εντολή βρίσκουμε τις MAC διευθύνσεις:

```
for PC in PC1 PC2 PC3 PC4; do VBoxManage showvminfo $PC | grep -o -e 'MAC: \w*' | xargs echo $PC; done
```

Οι MAC διευθύνσεις που προκύπτουν είναι οι εξής:

PC	MAC διεύθυνση
PC1	08:00:27:E6:40:E4
PC2	08:00:27:0C:12:D5
PC3	08:00:27:45:C3:01
PC4	08:00:27:A9:7D:C3

2.6 Στις B1, B2, B3: `ifconfig bridgeX flush` όπου $X = 1, 2, 3$ αντίστοιχα.

2.7 Στα PC1, PC2, PC3, PC4: `tcpdump -i em0 -e -n`

2.8 Στις B1, B2, B3: `ifconfig bridgeX addr` όπου $X = 1, 2, 3$ αντίστοιχα και βλέπουμε ότι δεν υπάρχουν εγγραφές.

Μπορούμε να πάμε σε 2ο παράθυρο είτε με το **tmux**, όπως στο ερώτημα 1.9 είτε με `<ALT> F2` να πάμε στο εικονικό τερματικό 2 (`ttyv2`).

Για να δούμε τους πίνακες προωθήσης εκτελούμε στις B1, B2, B3:
`ifconfig bridgeX addr` όπου $X = 1, 2, 3$ αντίστοιχα.

Η B1 έχει εγγραφές για τα PC1, PC2.

Η B2 έχει εγγραφές για τα PC1, PC2.

Η B3 έχει εγγραφές για το PC1.

2.9 Αρχικά το PC1 αποφασίζει ότι η 192.168.1.2 είναι στο ίδιο υπόδικτυο, οπότε στέλνει ARP Request για την IPv4 διεύθυνση 192.168.1.2. Το πλαίσιο αυτό έχει την broadcast MAC διεύθυνση FF:FF:FF:FF:FF:FF. Έτσι το προωθούν όλες οι γέφυρες. Οπότε από την διεύθυνση προέλευσης του πλαισίου αυτού, όλες οι γέφυρες μαθαίνουν σε ποια θύρα τους βρίσκεται το PC1.

Στην συνέχεια το PC2 απαντά με ARP Reply. Η γέφυρα B2 το ακούει αυτό, σημειώνει ότι το PC2 είναι στην θύρα που "βλέπει" στο LNK1, και δεν προωθεί το πλαίσιο αφού ήδη ξέρει ότι το PC1 βρίσκεται από την "πλευρά" της θύρας του LNK1 όπου ήδη βρίσκεται το πλαίσιο. Η γέφυρα B1 επίσης σημειώνει ότι το PC2 είναι στην θύρα που "βλέπει" στο LNK1, γνωρίζει ότι το PC1 είναι στην θύρα που "βλέπει" στο LAN1, οπότε προωθεί το πλαίσιο, το οποίο τελικά φθάνει στο LAN1 και άρα και στο PC1.

Στην συνέχεια το PC1 στέλνει το ICMP Echo Request προς το PC1. Και η B1 και η B2 ξέρουν που βρίσκεται το PC1 και το PC2 οπότε δεν μεταβάλλεται κάτι. Όμοια και όταν το PC2 στείλει το ICMP Echo Reply προς το PC2.

Συνοψίζοντας, όλες οι γέφυρες ξέρουν για το PC1, αλλά μόνο οι B1, B2 ξέρουν για το PC2.

2.10 Όχι. Οι B1 και B2 ξέρουν πού βρίσκεται το PC1 και το PC2 και επίσης το PC2 ξέρει την MAC διεύθυνση του PC1 (οπότε δεν στέλνει ARP Request). Συνεπώς, όπως στο προηγούμενο ερώτημα 2.9, δεν μεταβάλλεται κάτι.

2.11 Το PC2 στέλνει ARP Request για την IPv4 διεύθυνση 192.168.1.4 προς την broadcast MAC διεύθυνση FF:FF:FF:FF:FF:FF, οπότε το πλαίσιο αυτό το προωθούν όλες οι γέφυρες, οπότε φθάνει στο PC4. Έτσι και η B3 που δεν ήξερε για το PC2 μαθαίνει για το PC2. Στο πλαίσιο αυτό απαντά το PC4 με ARP Reply. Η B3 ξέρει ότι το PC2 είναι προς το LNK2 και έτσι προωθεί το πλαίσιο. Έπειτα η B2 προωθεί το πλαίσιο αυτό στο LNK1, αφού ξέρει ότι εκεί είναι το PC2.

TL;DR Το ARP Reply PC4 → PC2 βρίσκεται στο LNK1, οπότε το ακούει η B1 και έτσι σημειώνει ότι το PC4 είναι στην θύρα που "βλέπει" στο LNK1.

2.12 Όλες οι B1, B2, B3 έχουν εγγραφές για όλα τα PC1, PC2, PC3, PC4.

Για τα PC1, PC2, PC4 οι γέφυρες ξέρουν από προηγούμενα ερωτήματα. Τώρα, το PC3 στέλνει ARP Request προς την broadcast διεύθυνση FF:FF:FF:FF:FF:FF, που προωθείται από όλες τις γέφυρες. Έτσι, όλες οι γέφυρες μαθαίνουν και για το PC3.

2.13 PC4: `ping 192.168.1.2`

PC1: `ping 192.168.1.2`

2.14 Το ping PC4 → PC2 συνεχίζει να λειτουργεί φυσιολογικά. Αυτό συμβαίνει διότι PC4, PC2 βρέθηκαν στο ίδιο LAN, οπότε ανεξάρτητα του τι θα κάνουν οι γέφυρες, τα πλαίσια που εκπέμπει το PC4 πάνε απευθείας στο PC2 και αντίστροφα, οπότε η επικοινωνία μεταξύ τους λειτουργικά κανονικά.

2.15 Το ping PC1 → PC2 σταματάει να δείχνει replies, όμως δεν δείχνει και `Host is down`.

Αυτό που συμβαίνει είναι πως το PC1 έχει ήδη εγγραφή ARP για το PC2, οπότε συνεχίζει απλά να στέλνει μηνύματα ICMP Echo Request. Η γέφυρα B1 ορθά ξέρει ότι το PC2 είναι από την μεριά του LNK1, οπότε προωθεί το Echo Request από το LAN1 στο LNK1. Όμως, η γέφυρα B2, εξακολουθεί να πιστεύει από προηγουμένως, πως το PC2 βρίσκεται από την μεριά του LNK1, το οποίο, ως προς αυτήν, πια ειναι ψευδές. Έτσι, η B2 δεν προωθεί το πλαίσιο και έτσι το πλαίσιο δεν φθάνει ποτέ στο PC4.

2.16 Το ping PC1 → PC2 ξεκινά να λειτουργεί ξανά φυσιολογικά.

Το PC2 ξέρει την MAC διεύθυνση του PC3, οπότε στέλνει απευθείας ICMP Echo Request (χωρίς ARP Request). Το πλαίσιο φθάνει στην B3 που ξέρει πως το PC3 είναι προς το LNK2, οπότε το προωθεί εκεί. Έτσι η B2 ακούει αυτό το πλαίσιο, και μαθαίνει ότι το PC2 είναι από την μεριά του LNK2, και όχι του LNK1, όπως νόμιζε πριν, αλλάζοντας τον πίνακα προώθησής της.

Έτσι πια τα πλαίσια του ICMP Echo Request PC1 → PC2 η γέφυρα B2 τα προωθεί προς το LNK2 και μετά, με την σειρά της, η B3, επίσης τα προωθεί προς το LAN2, οπότε φθάνουν στο PC4. (η B3 έχει από την αρχή μάθει ότι το PC4 είναι προς το LAN2, λόγω των ICMP Echo Replies που το PC4 στέλνει προς το PC2, λόγω του ping PC2 → PC4).

2.17 Αν δεν συνέβαινε το ping αυτό, και αν δεν συνέβαινε καμία άλλη διαδικτυακή κίνηση, που να αποκάλυπτε την αλλαγή της τοπολογίας, κάποια στιγμή:

- Είτε θα έληγε η εγγραφή του πίνακα προώθησης της B2 για το PC2, οπότε θα έκανε broadcast τα πλαίσια με παραλήπτη αυτό,
- Είτε θα έληγε η ARP εγγραφή του PC1 για το PC2. Τότε PC1 θα στελνε προς την broadcast MAC διεύθυνση FF:FF:FF:FF:FF:FF ένα ARP Request, που λόγω του broadcast, θα έφτανε στο PC2. Το PC2 έπειτα θα έστελνε ARP Reply προς το PC1, που όλες οι γέφυρες ξέρουν την σωστή θέση του, οπότε θα έφτανε σε αυτό. Όμως καθώς θα μεταδιδόταν το ARP Reply θα περνούσε από την B2, που θα μάθαινε πια πως το PC2 είναι από την μεριά του LNK2, και όχι του LNK1.

Το 1ο ενδεχόμενο συμβαίνει στα 1200 δευτερόλεπτα (=20 λεπτά) από το τελευταίο μήνυμα με αποστολέα PC2 που ακούει η B2, όπως μπορούμε να δούμε και με `sysctl net.link.ether.inet.max_age`.

Το 2ο ενδεχόμενο συμβαίνει στα 1200 δευτερόλεπτα (=20 λεπτά) από το τελευταίο ARP μήνυμα που άκουσε που περιείχε την IPv4 και την MAC διεύθυνση του PC2.

Όποιο ενδεχόμενο συνέβαινε από αυτά πρώτο, θα αποκαθιστούσε την επικοινωνία PC1 ↔ PC2.

Άσκηση 3

3.1 Συγκρίνοντας τις MAC διευθύνσεις από το `ifconfig` στην B1 και από το VirtualBox, παρατηρούμε ότι στο LAN1 αντιστοιχεί η `em0` και στο LNK1 αντιστοιχεί η `em1`. Οπότε εκτελούμε στην B1:

```
ifconfig em0 up
ifconfig em1 up
ifconfig bridge1 create
ifconfig bridge1 addm em0
ifconfig bridge1 addm em1
ifconfig bridge1 up
```

3.2 Συγκρίνοντας τις MAC διευθύνσεις από το `ifconfig` στην B2 και από το VirtualBox, παρατηρούμε ότι στο LAN2 αντιστοιχεί η `em2` και στο LNK1 αντιστοιχεί η `em0`. Οπότε εκτελούμε στην B2:

```
ifconfig em0 up
ifconfig em2 up
bridge2 create
ifconfig bridge2 addm em0
ifconfig bridge2 addm em2
ifconfig bridge2 up
```

3.3 PC1: 08:00:27:E6:40:E4

PC2: 08:00:27:0C:12:D5

PC3: 08:00:27:45:C3:01

Εκτελούμε σε όλα τα PC1, PC2, PC3: `arp -a -d`

3.4 Στο PC1 βλέπουμε το ARP Request που έστειλε το PC2 προς την broadcast διεύθυνση για να μάθει την MAC διεύθυνση του PC3. Επειδή το πλαίσιο είναι προς την broadcast διεύθυνση, όλες οι γέφυρες το αναμετάδωσαν, οπότε έφτασε στο PC1. Στην απάντηση ARP Reply του PC2 (με αυτή η B2 μαθαίνει ότι το PC2 είναι στο LAN2) και έπειτα κατά την ICMP επικοινωνία PC2 ↔ PC3, η γέφυρα B2 ξέρει ότι τα PC2, PC3 βρίσκονται στο LAN2 οπότε δεν τα προωθει στο LNK1, και άρα δεν θα φτάνουν στο PC1.

3.5 Στο PC3:

```
ping 192.168.1.1
```

3.6 Στην B1:

```
ifconfig em2 up
ifconfig bridge1 addm em2
```

Στην B2:

```
ifconfig em1 up
ifconfig bridge2 addm em1
```

3.7 Η B1 προωθεί για το PC1 προς το `em0` (LAN1) και για τα PC2, PC3 προς το `em1` (LNK1). Η B2 προωθεί για το PC1 προς το `em0` (LAN2) και για τα PC2, PC3 προς το `em2` (LNK2).

3.8 Στην B1 το PC1 είναι στην `em0` και το PC3 στην `em1`.

Στην B2 το PC1 είναι στην `em0` και το PC3 στην `em2`.

3.9 Στα PC1, PC2: `tcpdump -i em0 -e -n`

3.10 Όχι.

3.11 Η ενέργεια που κάναμε είναι αποσύνδεση του LNK2 από την B1.

Αυτό που συμβαίνει και πακέτα ARP κατακλύζουν το δίκτυο, είναι πως πρόκειται για broadcast πλαίσιο οπότε όλες οι γέφυρες το προωθούν παντού. Έτσι η B2 το προωθεί προς την B1 μέσω και του LNK1 και του LNK2, η B2 προωθεί τα πλαίσια αυτά (αντίστροφα) προς την B1 μέσω των LNK2 και LNK1, αντίστοιχα, οπότε επαναλάμβνεται και αενάως ο κύκλος, καθώς η γέφυρα δεν έχει τρόπο να γνωρίζει ότι έχει ξαναστείλει αυτά τα πλαίσια.

Μια παρενέργεια είναι πως αφού αποσυνδέσουμε το LNK2 από την B1, ο κατακλυσμός μεν σταματά αφού ο κύκλος σπάει, όμως το τελευταίο πλαίσιο που λαμβάνει η B2 με αποστολέα το PC3 είναι από το LNK1 ή από το LNK2. Αυτό έχει ως αποτέλεσμα η γέφυρα B2, να καταγράψει, λανθασμένα, πως το PC3 είναι από την μεριά του LNK1 ή του LNK2.

Πράγματι τρέχοντας στην B2: `ifconfig bridge2 addr` παρατηρούμε πως η B2 έχει το PC3 στον πίνακα προώθησης, στην διεπαφή που "βλέπει" στο LNK1.

Αυτό έχει ως αποτέλεσμα το πλαίσιο ARP Reply που αποστέλλει το PC1 με παραλήπτη το PC3 να μην φτάσει ποτέ στο PC3, αλλά απλά να κινείται σε κύκλους μεταξύ LNK1 και LNK2.

3.12 Ερώτηση ARP (PC3 → broadcast): Ποιος έχει την διεύθυνση 192.168.1.1, πείτε στον 192.168.1.3 (με παραλήπτη την MAC του PC3);

Απάντηση ARP (PC1 → PC3): Την 192.168.1.3 έχει η MAC διεύθυνση `[PC3 MAC ADDRESS]`.

3.13 PC3

3.14 Απαντήθηκε στο 3.11

3.15 Απαντήθηκε στο 3.11

Άσκηση 4

4.1 Στις B1, B2:

```
ifconfig bridgeX destroy
ifconfig em0 down
ifconfig em1 down
ifconfig em2 down
```

`ifconfig bridgeX create`, όπου $X = 1, 2$ αντίστοιχα.

4.2 Στην B1:

```
ifconfig em0 up
ifconfig em1 up
ifconfig em2 up
ifconfig lagg0 create
```

4.3 Στην B1:

```
ifconfig lagg0 laggport em1 laggport em2 up
```

4.4 Στην B2:

```
ifconfig em0 up
ifconfig em1 up
ifconfig em2 up
ifconfig lagg0 create
ifconfig lagg0 laggport em0 laggport em1 up
```

4.5 Στην B1:

```
ifconfig bridge1 addm em0 addm lagg0 up
```

4.6 Στην B2:

```
ifconfig bridge2 addm em2 addm lagg0 up
```

4.7 Στο PC1:

```
tcpdump -i em0 -e -n
```

Με ping PC3 → PC2, στο LAN1 εμφανίζεται μόνο το σχετικό ARP Request που κάνει broadcast το PC3 για να μάθει την MAC διεύθυνση του PC2. Αφού εκπεμπφθεί αυτό, και αφού το PC3 στείλει το ARP Reply, η B2 γνωρίζει ότι τα PC2, PC3 είναι και τα δύο στο LAN2, οπότε δεν προωθεί την κίνηση μεταξύ τους.

4.8 Στο PC1:

```
tcpdump -i em0 -e -n
```

4.9 Στο PC3:

```
arp -a -d
```

Ναι το ping επιτυγχάνει.

4.10 Τα πακέτα ICMP εμφανίζονται μόνο στην σύνδεση LNK1. Αυτό συμβαίνει διότι όπως βλέπουμε με την `ifconfig lagg0` στις B1, B2, το πρώτοκολλο συνάθροισης της διεπαφής συνάθροισης είναι `failover`, που σημαίνει ότι χρησιμοποιείται μία μόνο ζεύξη στην κανονική λετουργία και κάποια άλλη μπαίνει σε λειτουργία, μόνο αν η κανονικά λειτουργική σταματήσει να λειτουργεί.

4.11 Το ping επιτυγχάνει. Τα πλαίσια πλέον μεταφέρονται μέσω της ζεύξης LNK2, αντί της LNK1.

4.12 Το ping επιτυγχάνει ξανά, όμως τα πλαισία πλέον μεταφέρονται ξανά μέσω της ζεύξης LNK1, όπως αρχικά.

Άσκηση 5

5.1 Στις B1, B2:

```
ifconfig bridgeX destroy
ifconfig lagg0 destroy
ifconfig em0 down
ifconfig em1 down
ifconfig em2 down
```

όπου $X = 1, 2$ αντίστοιχα.

5.2 Στην B1:

```
ifconfig em0 up
ifconfig em1 up
ifconfig em2 up
ifconfig bridge1 create
ifconfig bridge1 addm em0 addm em1 addm em2 up
```

5.3 Στην B2:

```
ifconfig em0 up
ifconfig em1 up
ifconfig em2 up
ifconfig bridge2 create
ifconfig bridge2 addm em0 addm em1 addm em2 up
```

5.4 Στην B1: `ifconfig bridge1 stp em0 stp em1 stp em2`

5.5 Στην B2: `ifconfig bridge2 stp em0 stp em1 stp em2`

5.6 Στις B1, B2 τρέχουμε `ifconfig bridgeX`, $X = 1, 2$ και βλέπουμε ότι η B1 έχει id 08:00:27:54:20:aa και priority 32768 και η B2 έχει id 08:00:27:2e:07:60 και priority 32768. Συνεπώς το Bridge ID της B1 είναι (32768, 08:00:27:54:20:aa) και το Bridge ID της B2 είναι: (32768, 08:00:27:2e:07:60).

Προβληματίζομαστε σχετικά με τον τρόπο επιλόγης της Bridge MAC Address για την επιλογή στο Bridge ID = (priority, Bridge MAC Address). Μια γέφυρα δεν έχει μια διεπαφή δικτύου, αλλά τουλάχιστον 2, οπότε το Bridge MAC Address δεν είναι σαφές ποια τιμή πρέπει να λάβει. Στο FreeBSD παρατηρούμε πως γίνεται assign μιας τυχαίας MAC διεύθυνσης στην γέφυρα, που όμως δεν επιλέγεται ως Bridge ID (από μελέτη στο διαδίκτυο για να μην συμβούν MAC address conflicts). Αυτό που φαίνεται να ισχύει είναι πως το FreeBSD επιλέγει ως Bridge MAC Address την έλαχιστη MAC address (σε big-endian σύγκριση), αν και αυτό δεν φαίνεται να ορίζεται κάπου συγκεκριμένα.

5.7 Σύμφωνα με το `ifconfig` και στις δύο B1, B2, η B2 είναι η γέφυρα-ρίζα. Πράγματι η Bridge ID της B2 είναι μικρότερη της Bridge ID της B1.

5.8 Με `ifconfig` στην γέφυρα-ρίζα B2 παρατηρούμε πως όλες οι διεπαφές είναι Designated (επιλεγμένες), όπως αναμένεται να συμβαίνει στην γέφυρα-ρίζα.

5.9 Η `em1` που "βλέπει" στο LNK1.

5.10 Η άλλη διεπαφή που να βλέπει στα LNK1/LNK2, `em2`, της μη ριζικής γέφυρας, B1, έχει ρόλο alternate και βρίσκεται σε κατάσταση discarding.

5.11 Η διεπαφή που βλέπει στο LAN1, `em0`, της μη ριζικής γέφυρας, B1, έχει ρόλο designated και βρίσκεται σε κατάσταση forwarding.

5.12 Η γέφυρα-ρίζα έτυχε να είναι η B2 και όχι η B1, οπότε καταγράφω την κίνηση στο LAN2 τρέχοντας στην B2:

```
tcpdump -i em2 -e -n -vv
```

BPDU εκπέμπεται κάθε 2 δευτερόλεπτα.

5.13 IEEE 802.3

5.14 Διεύθυνση MAC πηγής είναι η διεύθυνση της διεπαφής `em2` της B2, που "βλέπει" στο LAN2 (διαφορετική από την Bridge MAC Address για το Bridge ID της B2)

Διεύθυνση MAC προορισμού είναι η: 01:80:C2:00:00:00. Η διεύθυνση αυτή έχει 1 στο LSB του 1ου byte, οπότε είναι μια διεύθυνση multicast (πολυεκπομπής). Σύμφωνα με το άρθρο της Wikipedia https://en.wikipedia.org/wiki/Bridge_protocol_data_unit τα BPDUs αποστέλλονται multicast προς αυτή την διεύθυνση ώστε να τα λαμβάνουν όλες οι γέφυρες, άλλα μόνο αυτές, δηλαδή όχι οι υπόλοιποι hosts ενός δικτύου.

5.15 Στην διεπαφή `em2`, που βλέπει στο LAN2, όπου και καταγράφουμε την δικτυακή κίνηση.

5.16 Mutlicast, καθώς έχει 1 στο LSB του 1ου byte.

5.17 Root ID: `8000.08:00:27:2e:07:60`

Bridge ID: `8000.08:00:27:2e:07:60.8003`

Root Path Cost: 0

5.18 Το 1ο καθώς το 3ο μέρος άλλαξε. Εξάλλου με `ifconfig` παρατηρούμε κιόλας πως το priority είναι 32768=0x8000.

5.19 Η Bridge MAC Address. Βλ. σχόλιο ερώτησης 5.6 σχετικά με το πώς φαίνεται να επιλέγεται αυτή.

5.20 Κάθε γέφυρα αριθμεί τις θύρες της αναθέτωντας σε κάθε μια ένα Port ID. Το Port ID αυτό χρησιμοποιείται ως τελευταίο κριτήριο επίλυσης για την επιλογή ριζικής θύρας. Συγκεκριμένα, το κριτήριο αυτό είναι αναγκαίο όταν δύο γέφυρες συνδέονται με παραπάνω από 1 συνδέση.

5.21 Όχι. Μια γέφυρα διαφημίζει τον εαυτό της με BPDU, μόνο όταν προσφέρει καλύτερο κόστος προς την ρίζα, οπότε στην σύνδεση μη ριζικής γέφυρας - ριζικής γέφυρας, η ριζική γέφυρα έχει 0 κόστος προς την ρίζα, ενώ η άλλη μη μηδενικό.

5.22 Στην `em0` που βλέπει στο LAN1.

5.23 Root ID: `8000.08:00:27:2e:07:60`

Bridge ID: `8000.08:00:27:54:20:aa.8001`

Root Path Cost: 20000

5.24 Ναι είναι.

5.25 Πέρνανε 7 δευτερόλεπτα. Ναι είναι σύμφωνος. Το RSTP προβλέπει χρόνο $3 \cdot$ `HELLOTIMES` με το default hello time να είναι 2 δευτερόλεπτα (επιβεβαιώνουμε με `ifconfig` στην B2 πως το hellotime είναι πράγματι 2 δευτερόλεπτα). Άρα περνάνε θεωρητικά 6 δευτερόλεπτα μέχρι η επικοινωνία να αποκασταθεί. Ο λόγος που χρειάζονται 7 δευτερόλεπτα αντί 6, είναι διότι η επικοινωνία αποκαθίσταται στα 6, αλλά το ping στέλνει ανά 1 δευτερόλεπτο, οπότε πρέπει να φθάσει ο χρόνος της επόμενης εκπομπής μετά την αποκάτασταση.

5.26 Όχι, δεν υπάρχει διακοπή στην επικοιωνία. Η αλλαγή πίσω στο LNK1 γίνεται στιγμιαία χωρίς να το καταλάβουν οι hosts. (η αλλαγή συμβαίνει εκ νέου λόγω του κανόνα επίλυσης που επιβάλλει ριζική θύρα να είναι εκείνη με το μικρότερο Port ID).

Άσκηση 6

6.1 Στην B1:

```
ifconfig em3 up
ifconfig bridge1 addm em3 stp em3
```

6.2 Στην B2:

```
ifconfig em3 up
ifconfig bridge2 addm em3 stp em3
```

6.3 Στην B3:

```
ifconfig em0 up
ifconfig em1 up
ifconfig em2 up
ifconfig em3 up
ifconfig bridge3 create
ifconfig bridge3 addm em0 addm em1 addm em2 stp em0 stp em1 stp em2 up
```

6.4 Στις B1, B2, B3: `ifconfig bridgeX flush` όπου $X = 1, 2, 3$ αντίστοιχα.

Ναι τα ping PC1 → PC2 και PC1 → PC3 είναι επιτυχή.

6.5 Στην B1: `ifconfig bridge1 priority 32767`

6.6 Με `ifconfig` βλέπουμε ότι το path cost και των τριών είναι 20000. Πράγματι, επίσης από το `ifconfig` βλέπουμε ότι οι αντίστοιχες διεπαφές δικτύου είναι στα 1000 mbps, εύρος ζώνης που αντιστοιχεί σε τιμή path cost 20000.

6.7 Με `tcpdump -i em0 -vv` στην B3 παρατηρούμε ότι η B3 λαμβάνει root path cost από την B1 ίσο με 0, που είναι λογικό αφού η B1 είναι η ρίζα.

Με `tcpdump -i em1 -vv` στην B3 παρατηρούμε ότι η B3 λαμβάνει root path cost από την B3 ίσο με 20000, που είναι λογικό αφού η B1 απέχει με μονοπάτι ελαχίστου κόστους 1000 από την B1 (η απευθείας ακμή).

6.8 Με `ifconfig` στην B3 παρατηρούμε ότι η ριζική θύρα της B3 είναι η `em0`, δηλαδή εκείνη που "βλέπει" στο LNK3.

Πράγματι και η B1 και η B2 διαφημίζουν ως γέφυρα-ρίζα την B1, οπότε η ριζική θύρα καθορίζεται από την θύρα που έχει το μικρότερο κόστος προς την ρίζα. Αυτή η θύρα είναι η `em0`, που "βλέπει" στο LNK3, καθώς έχει κόστος 20000, ενώ η άλλη θύρα έχει κόστος 20000+20000=40000.

6.9 Στην B3 η μη ριζική θύρα `em1`, που βλέπει στο LNK4, παρατηρούμε ότι είναι με ρόλο επιλεγμένη και σε κατάσταση forwarding. Αντίθετα, στην B2 η θύρα που βλέπει στο LNK4, παρατηρούμε ότι είναι με ρόλο alternate και σε κατάσταση discarding.

Αυτό συμβαίνει (η B3 να γίνει η επιλεγμένη γέφυρα για το LNK4) διότι:

- Οι B2, B3 έχουν ίδια αντίληψη για την ριζική ρίζα˙ αυτή είναι η B1.
- Οι B2, B3 ίδιο κόστος μονοπατιού προς την ρίζα.
- Η B3 έχει μικρότερο Bridge ID από την B2 (ίδιο priority αλλά MAC B3 = 08:00:27:08:3d:87 < MAC B2 = 08:00:27:2e:07:60).

6.10 Με `tcpdimp -i em2 -e -vv` παρατηρούμε πως το root path cost είναι 20000. Πράγματι, το μονοπάτι ελάχιστου κόστους της B3 προς την ρίζα είναι το B3 → B1 κόστους 20000.

6.11 Στο PC1: `ping 192.168.1.3`

6.12 Στην B3: `ifconfig bridge3 ifpathcost em0 40001`.

Θέσαμε το κόστος της ακμής B3-B1 σε μια μονάδα πάνω από το κόστος του ελάχιστου μονοπάτιου B3-B1 αγνοώντας την ακμή B1-B3, που είναι το B3-B2-B1, κόστους 20000+20000=40000.

6.13 Η επικοινωνία αποκαταστάθηκε μετά από 2 secs, δηλαδή σε χρόνο ίσο με 1 hello time.

6.14 Η διεπαφή `em0` της bridge3 της B3 είναι σε ρόλο Alternate και σε κατάσταση Discarding. (αυτό συμβαίνει διότι για το LNK3 η επιλεγμένη γέφυρα είναι η B1 με κόστος προς την ρίζα 0)

Η διεπαφή `em2` της bridge2 της B2 είναι σε ρόλο Designated και σε κατάσταση Forwarding. (αυτό συμβαίνει διότι για το LNK4 η επιλέγμνη γέφυρα είναι η B2 με κόστος προς την ρίζα 20000 – η B3 έχει κόστος 40001)

6.15 Ναι. Στο LNK4 προηγουμένως, εξέπεμπε η B3, ενώ πλέον εκπέμπει η B2. Κατά τα άλλα (εξαιρώντας τις MAC διευθύνσεις) η B2 εκπέμπει ό,τι εξέπεμπε πριν η B2.

6.16 Ναι. Στο LAN3 η B3 πια διαφημίζει κόστος προς την ρίζα 40001 αντί για 20000 που διαφήμιζε πριν.

6.17 8 δευτερόλεπτα

6.18 4 δευτερόλεπτα

6.19 Η μία είναι σε ρόλο Designated και κατάσταση Forwarding, ενώ η άλλη είναι σε ρόλο Backup και κατάσταση Discarding.

6.20 Με το προεπιλεγμένο κόστος (20000) η διεπαφή `em0` της B3 στο LNK3 ήταν η ριζική θύρα της B3. Αρκεί συνεπώς απλά να επαναφέρουμε προεπιλεγμένο κόστος. Έτσι στην B3:

```
ifconfig bridge3 ifpathcost em0 0
```

(το 0 επαναφέρει την προεπιλεγμένη τιμή, δηλαδή το 20000)

Άσκηση 7

7.1 Στο PC1:

```
ifconfig em0.5 create 192.168.5.1/24 up
ifconfig em0.6 create 192.168.6.1/24 up
```

7.2 Στην B1:

```
ifconfig em0.5 create up
ifconfig em0.6 create up
```

7.3 Στην B1:

```
ifconfig em1.6 create up
ifconfig em3.5 create up
```

7.4 Στο PC2:

```
ifconfig em0.6 create 192.168.6.2/24 up
```

7.5 Στην B2:

```
ifconfig em3.6 create up
ifconfig em0.6 create up
```

7.6 Στο PC3:

```
ifconfig em0.5 create 192.168.5.3/24 up
```

7.7 Στην B3:

```
ifconfig em2.5 create up
ifconfig em0.5 create up
```

7.8 Ναι επιτυγχάνει.

7.9 Αντιλαμβανόμαστε ότι ζητείται να να απενεργοποιήσουμε το STP στην διεπαφή της B1 στο LAN1, δηλαδή να καταστήσουμε την διεπαφή αυτή μόνιμη στο επικαλύπτον δέντρο. (διαφορετικά αν την βγάλουμε από το επικαλύπτον δέντρο το PC1 θα αποσυνδεθεί από το δίκτυο). Τότε στο B1:

```
ifconfig bridge1 -stp em0
```

7.10 Στο PC1:

```
tcpdump -i em0 -n -xx -vv -e
```

7.11 Στο PC2:

```
arp -a -d
```

Το Ethertype έχει τιμή 0x0800 για το IPv4 και τιμή 0x0806 για το ARP. (στο Ethernet II)

7.12 Έχουν Ethertype με τιμή 0x8100 που σημαίνει 802.1Q. Ακολουθεί, πρακτικά στην θέση του payload, δηλαδή πρακτικά ενθαλυκωμένο το πλαίσιο 802.1Q, το οποίο αποτελείται αρχικά από 4 control bits, μετά 12 bits για το VLAN tag και στο τέλος το κανονικό ενθυλακωμένο πλαίσιο, όπως θα ήταν αν δεν υπήρχε η 802.1Q επικεφαλίδα. (δηλαδή ακολουθεί το Ethertype, έπειτα το Payload)

7.13 Το Ethertype της Ethernet επικεφαλίδας είναι 0x8100 που σημαίνει 802.1Q, το Ethertype που υπάρχει αμέσως μετά, δηλαδή στην επικεφαλίδα 802.1Q, είναι 0x0800 για το IPv4 και 0x0806 για το ARP.

7.14 Στα 12 τελευταία bits του 3ου και 4ου byte της 802.1Q επικεφαλίδας, που βρίσκεται ακριβώς μετά την διεύθυνση MAC παραλήπτη.

7.15 Στο PC1:

```
tcpdump -i em0.5 -n -xx -vv -e
```

7.16 Στο PC6:

```
arp -a -d
```

Το Ethertype είναι 0x0800 για το IPv4 (ενθυλακώνουν τα μηνύματα ICMP εδώ) και 0x0806 για το ARP.

Πεδίο σχετικό με το VLAN δεν υπάρχει. Το FreeBSD έχει αφαιρέσει την επικεφαλίδα 802.1Q.

7.17 Στο PC1:

```
tcpdump -i em0 -n -xx -vv -e
```

7.18 Τα πλάσια BPDU είναι φυσιολογικά πλαίσια Ethernet 802.3 χωρίς επικεφαλίδα 802.1Q. Είναι, εξάλλου, λογικό, αφού στην γέφυρα `bridge1` της B1 έχουμε προσθέσει την διεπαφή `em0`, οπότε η γέφυρα δεν γνωρίζει για VLANs.

Ωστόσο, διαφέρουν από τα προηγούμενα πλαίσια, στο γεγονός ότι είναι πλαίσια 802.3 αντί Ethernet II, οπότε αντί του πεδίου Ethertype έχουν το πεδίο Length, που είναι το μέγεθος του payload σε bytes. (Ο διαχωρισμός μεταξύ 802.3 και Ethernet II γίνεται καθώς το μέγιστο μέγεθος payload είναι 1500 bytes, οπότε αν η τιμή του Ethertype/Length είναι μικρότερη ή ίση του 1500, τότε πρόκειται για πλαίσιο 802.3, ενώ αλλιώς πρόκειται για πλαίσιο Ethernet II)

7.19 Στο PC1:

```
tcpdump -i em0 -n -xx -vv -e 'not stp'
```