Ονοματεπώνυμο: Ανδρέας Στάμος (03120***)	**Όνομα PC:** linux / Ubuntu 22.04.2 LTS
Ομάδα: 1	**Ημερομηνία:** 12/03/2024

Εργαστηριακή Άσκηση 5
Στατική δρομολόγηση

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 Στο PC1:

```
ifconfig em0 192.168.1.2/24 up
```

Στο PC2:

```
ifconfig em0 192.168.2.2/24 up
```

1.2 Στο R1:

```
sysrc ifconfig_em0="192.168.1.1/24" ifconfig_em1="192.168.2.1/24"
```

1.3
```
gateway_enable="YES"
```

1.4 Στο R1:

```
service netif restart && service routing restart
```

1.5 Στο PC1:

```
route add 192.168.2.0/24 192.168.1.1
```

1.6 Στο PC1:

```
netstat -rn
```

Οι σημαίες (flags) είναι `UGS`, όπου το `U` σημαίνει πως η διαδρομή είναι ενεργή, το `G` σημαίνει πως μεσολαβούν δρομολογητές και το `S` σημαίνει πως πρόκειται για διαδρομή που έχει οριστεί στατικά.

1.7 Δεν βλέπουμε Echo Replies, όμως ούτε και κάποιο μήνυμα λάθους.

1.8 Στον R1:

```
tcpdump -i em0 -e -n
tcpdump -i em1 -e -n
```

Μηνύματα ICMP Echo Request εκπέμπονται από το PC1 στο LAN1, ο R1 τα μετάγει από τον LAN1 στο LAN2, οπότε φτάνουν στο PC2. Όμως, το PC2 δεν έχει κανόνα δρομολόγησης για την διεύθυνση 192.168.1.2 του PC1, οπότε δεν στέλνει Echo Reply.

1.9 Στο PC2:

```
route add 192.168.1.0/24 192.168.2.1
```

1.10 Ναι το ping επιτυγχάνει.

1.11 Για τον R1, οι διευθύνσεις 192.168.1.2 και 192.168.2.2 είναι απευθείας προσβάσιμες από τις διεπάφες του `em0` και `em1` αντίστοιχα (καθώς τους έχουν διευθύνσεις αντίστοιχα 192.186.1.1/24 και 192.168.2.1/24). Από προεπιλογή, υπάρχουν στον πίνακα δρομολόγησης οι απευθείας διαδρομές που προκύπτουν από τον αριθμό δικτύου κάθε διεπαφής.

Άσκηση 2

2.1 Στο PC1:

```
route delete 192.168.2.0/24
```

2.2 Στο PC1:

```
ifconfig em0 192.168.2.1/20
```

2.3 Στο ίδιο μαζί του, καθώς θεωρεί πως ανήκουν και τα 3 PCs στο 192.168.0.0/20

2.4 Όχι αποτυγχάνουν και με μήνυμα λάθους `Host is down`. Αυτό συμβαίνει καθώς το PC1 θεωρεί ότι τα PC2, PC3 βρίσκονται στο ίδιο υποδίκτυο μαζί του, οπότε εκδίδει ARP Request για τις IPv4 διευθύνσεις τους, που δεν το απαντάει κανένας.

2.5 Επιτυγχάνει καθώς το PC1 θεωρεί ότι η IPv4 διεύθυνση του PC2 ανήκει στον R1 (αφού το R1 εξέδοσε σχετικό ARP Reply), οπότε στέλνει τα πλαίσια προς αυτόν, ο οποίος τα προωθεί στο PC2. Στην αντίστροφη πορεία, όμως, το PC2, στέλνει τα πακέτα Echo Replies, συνειδητά, προς τον R1, που τα προωθεί στο PC1. Η διαφορά μεταξύ των δύο πορειών βρίσκεται στο γεγονός πως το PC1 θεωρεί ότι το PC2 είναι στο υποδίκτυο του και πως του προωθεί απευθείας πλαίσια, ενώ το PC2 θεωρεί πως το PC1 δεν βρίσκεται στο υποδίκτυο του, οπότε πρέπει να προωθήσει τα πακέτα προς κάποιον δρομολογητή.

2.6 Αποτυχγάνει, χωρίς όμως κάποιο μήνυμα λάθους. Τα Echo Requests φθάνουν στο PC3 το οποίο όμως δεν έχει κανόνα δρομολόγησης προς την IPv4 διεύθυνση 192.168.1.2 του PC1, οπότε δεν μπορεί να στείλει Echo Reply.

2.7 Στο PC3:

```
route add 192.168.1.0/24 192.168.2.1
```

2.8 Στα PC1, PC3:

```
arp -a -d
```

2.9 Στον R1:

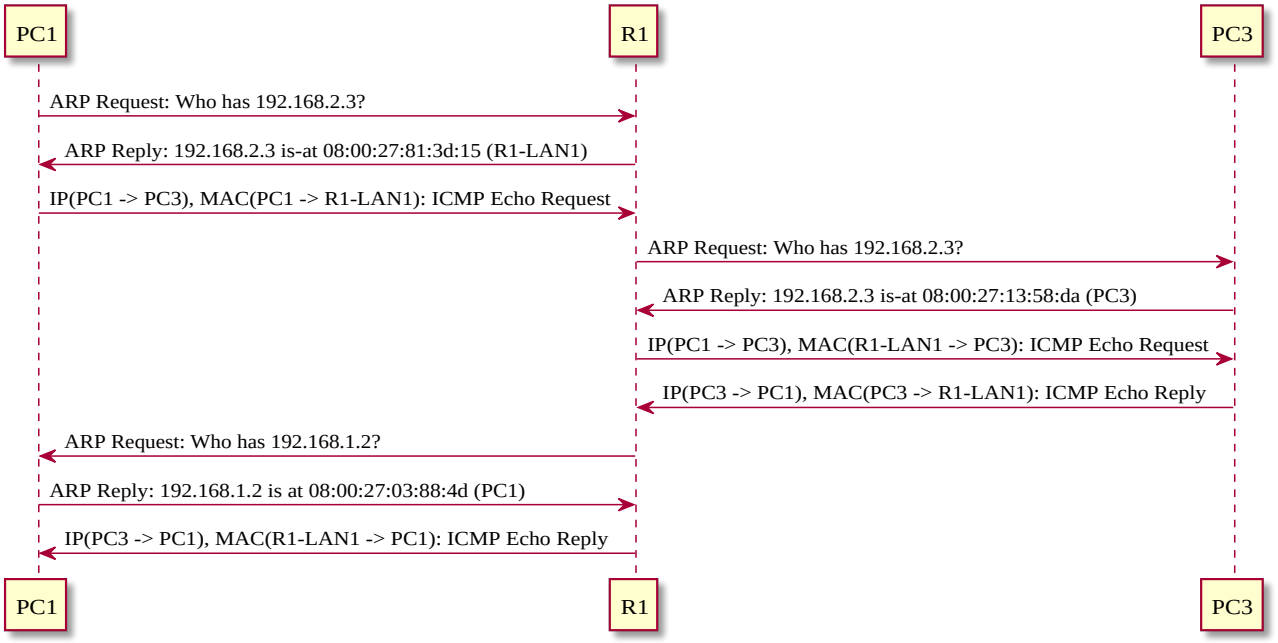
```
tcpdump -i em0 -e -n
tcpdump -i em1 -e -n
```

2.10 Το R1 απάντησε στο ARP Request του PC1 που στην IPv4 διεύθυνση του PC3 αντιστοιχεί η MAC διεύθυνση της διεπαφής του στο LAN1.

2.11 Προς την διεπαφή του R1 στο LAN1.

2.12 Από την διεπαφή του R1 στο LAN2.

2.13



Σημείωση: Παρατηρούμε το πάραδοξο πως ο R1 ενώ έχει ακούσει από το ARP Request την MAC διεύθυνση του PC1 (δηλ. της IPv4 διεύθυνσης 192.168.2.1), όταν είναι να στείλει ARP Reply στέλνει ARP Request για την διεύθυνση αυτή, δηλαδή δεν αποθήκευσε στην ARP cache την αντιστοιχία αυτή. Ο λόγος που συμβαίνει

αυτό, είναι καθώς το RFC 826 ορίζει πως μια NEA αντιστοιχία δημιουργείται στον κρυφή μνήμη ARP, μόνο αν το ARP πλαίσιο έχει διεύθυνση πρωτοκόλλου (IPv4) παραλήπτη την διεύθυνση του host που το λαμβάνει (αυτό ισχύει αν πρόκειται για νέα εγγραφή, αν ήδη υπάρχει, γίνεται ανανένωση της πριν γίνει ο έλεγχος αυτός). Στην συγκεκριμένη περίπτωση αυτό δεν ισχύει, καθώς το ARP πλαίσιο έχει διεύθυνση πρωτοκόλλου παραλήπτη την διεύθυνση IPv4 του PC3.

2.14 Αρκεί και πρέπει το PC1 να θεωρεί ότι το PC3 είναι στο ίδιο υποδίκτυο με αυτό, δηλαδή να έχουν οι διευθύνσεις τους ίδιο αριθμό δικτύου, με βάση το πρόθεμα που θα τεθεί. Το μέγιστο κοινό πρόθεμα των δύο διευθύνσεων 192.168.1.2 και 192.168.2.3 είναι 22 bits μήκους. Συνεπώς, η μέγιστη τιμή μήκους προθέματος, ώστε το ping αυτό να λειτουργεί, είναι 22.

2.15 Στο PC1:

```
ifconfig em0 192.168.2.1/23
```

2.16 Στο PC1:

```
route add 192.168.2.0/24 -interface em0
```

2.17 `link#1`

2.18 Ναι επιτυγχάνει.

Συμβαίνει ότι και προηγουμένως. Το PC1 θεωρεί ότι το PC3 είναι στο ίδιο υποδίκτυο με αυτό, οπότε εκδίδει ARP Request για αυτό. Στην συνέχεια, ο R1, λόγω της λειτουργίας του ως ARP Proxy, απαντά στο ARP Request στέλνοντας την δική του MAC διεύθυνση όμως. Έτσι, έπειτα, το PC1 στέλνει το πακέτο Echo Request προς τον R1 θεωρώντας πως το στέλνει στο PC3. Ο R1 προωθεί το πακέτο κανονικά προς το PC3 (αφού πρώτα κάνει σχετικό ARP Request). Τέλος, το PC3 στέλνει Echo Reply με διεύθυνση IPv4 παραλήπτη το PC1. Το πακέτο αυτό στέλνει προς τον δρομολογητή R1, ο οποίος αφού μάθει την MAC διεύθυνση του PC1 μέσω ARP (αυτό απαιτείται για τον λόγο που περιγράφεται στην **Σημείωση** του ερωτήματος 2.13), στέλνει τελικά το πακέτο προς το PC1.

2.19 Στον R1:

```
sysctl net.link.ether.inet.proxyall=0
```

2.20 Στον R1:

```
route change 192.168.2.0/24 192.168.1.1
```

2.21 Στο PC1:

```
ifconfig em0 192.168.1.2/24
```

2.22 Έχει διαγραφεί.

2.23 Στον R1:

```
route add 192.168.2.0/24 192.168.1.1
```

2.24 Στο PC3:

```
ifconfig em0 delete
```

Η αποσύνδεση γίνεται από τα Settings του VirtualBox.

Άσκηση 3

3.1 Στον R1:

```
sysrc ifconfig_em1="172.17.17.1/30"
service restart netif
```

3.2 Στον R2:

```
sysrc ifconfig_em0="172.17.17.2/30" ifconfig_em1="192.168.2.1/24"
service restart netif
```

3.3 Στον R2:

```
sysrc gateway_enable="YES" && service restart routing
```

3.4 Λαμβάνουμε ICMP Destination host unreachable από τον R1.

3.5 Στο LAN1 παράγονται ICMP Echo Requests από το PC1 και ICMP Host Unreachable από τον R1. Στο WAN1 δεν παράγονται μηνύματα.

Αυτό που συμβαίνει, είναι ότι αρχικά το PC1 θεωρεί ότι για να στείλει στο 192.168.2.2 πρέπει να προωθήσει το πακέτο στον R1 στο 192.168.1.1. Ο R1 λαμβάνοντας το πακέτο, βλέπει ότι δεν έχει κανόνα δρομολόγησης για την διεύθυνση 192.168.2.2 και έτσι απορρίπτει το πακέτο, στέλνοντας και σχετικό μήνυμα λάθους ICMP Host Unreachable προς τον αποστολέα, που είναι το PC1.

3.6 Παρατηρούμε ένα hop από τον 192.168.1.1 και επίσης ένα δεύτερο μήνυμα από τον 192.168.1.1 με την επισημείωση `!H`, που από το σχετικό man page, παρατηρούμε πως σημαίνει Host Unreachable.

3.7 Στον R1:

```
route add 192.168.2.0/24 172.17.17.2
```

3.8 Αποτυγχάνει χωρίς όμως κάποιο μήνυμα λάθους.

3.9 Στο LAN2 εκπέμπεται:

1. Από τον R1 το προωθημένο UDP πακέτο που έχει IP αποστολέα το PC1.
2. Από το PC2 ένα ICMP Port Unreachable με IP παραλήπτη το PC1 και MAC παραλήπτη τον R2.

Αυτό που συμβαίνει είναι πως το πακέτο Echo Request το PC1 το στέλνει προς τον R1, ο R1 το προωθεί στον R2, που τελικά το προωθεί στο PC2. Στην συνέχεια, το PC2 αποστέλλει ένα Echo Reply προς τον R2, όμως ο R2 δεν έχει κανόνα δρομολόγησης για την διεύθυνση 192.168.1.2 του PC1 και έτσι απορρίπτει το πακέτο. Θεωρητικά θα έπρεπε να στείλει ένα ICMP Host unreachable προς το PC2, όμως επειδή θα επρόκειτο για μήνυμα ICMP λάθους που εκδίδεται σε απάντηση ενός μηνύματος ICMP λάθους (το Port Unreachable) δεν αποστέλλεται το ICMP Host Unreachable προς το PC2.

3.10 Το traceroute δεν ολοκληρώνεται. Παρατηρούμε το πρώτο hop που είναι στην διεύθυνση 192.168.1.1, δηλαδή από τον R1 και στην συνέχει τα επόμενα hops είναι `* * *`, δηλαδή το traceroute δεν έλαβε κάποιο μήνυμα οπότε συνέχισε να προσπαθεί να στέλνει πακέτα για να ανακαλύψει την διαδρομή με μεγαλύτερο TTL (με την λογική πως η διαδρομή στα αλήθεια θα ήταν μεγαλύτερη, αλλά απλά κάποιοι ενδιάμεσοι κόμβοι απορρίπτουν μηνύματα χωρίς να απαντάνε).

Αυτό, όμως που συμβαίνει, όπως εξηγήσαμε και στην ερώτηση 3.9, είναι πως υπάρχουν κανόνες δρομολόγησης για την διαδρομή PC1 → R1 → R2 → PC2, όμως στην αντίστροφη διαδρομή, λείπει ο κανόνας R2 → PC1[1], οπότε ο R2 αδυνατεί να στείλει οποιοδήποτε μήνυμα προς το PC1 είτε πρόκειται για το Echo Reply του PC2, είτε πρόκειται για το ICMP Time Exceeded που θα έπρεπε να στείλει προς το PC1, λόγω της απόρριψης το με μηδενικό TTL πακέτου Echo Request που είχε στείλει το PC1 (αυτό το πακέτο είχε αρχικά TTL=2).

3.11 Παράγονται τα προωθημένα UDP πακέτα με IP αποστολέα το PC1 και ICMP Port Unreachable μηνύματα με αποστολέα το PC2.

3.12 Θεωρητικά ο R2 όταν άπερριψε το πακέτο ICMP Port Unreachable επειδή δεν είχε κανόνα δρομολόγησης για την διεύθυνση 192.168.1.2 του PC1, έπρεπε να στείλει ICMP Host Unreachable προς το PC2. Όμως το ICMP Port Unreachable είναι μήνυμα ICMP λάθους, και μηνύματα ICMP λάθους για πακέτα ICMP λάθους δεν αποστέλλονται.

Συμπλήρωση: Αν είχαμε εκτελέσει το traceroute με την επιλογή `-I` ώστε να χρησιμοποιήσει ICMP Echo Request, αντί για UDP προς τυχαία θύρα, τότε το PC2 θα είχε εκδόσει ICMP Echo Reply, που είναι πληροφοριακό μήνυμα ICMP και όχι λάθους. Στην περίπτωση αυτή, και πράγματι δοκιμάστηκε και ισχύει, το R2 θα εξέδιδε πράγματι ICMP Host Unreachable προς το PC2.

3.13 Στον R2:

```
route add 192.168.1.0/24 172.17.17.1
```

3.14 Ναι επιτυγχάνει.

Στο WAN1 παράγονται δεδομενογράμματα UDP που προωθεί ο R1 από το PC1 προς τον R2 και επίσης μηνύματα ICMP Time Exceeded που εκδίδει ο R2 όταν μηδενιστεί σε αυτόν το TTL των πακέτων που στέλνει ο PC1 (αυτό θα συμβεί για πακέτα με αρχικό TTL=2)

3.15 Το ping αποτυγχάνει με μήνυμα λάθους `No route to host`.

Αυτό συμβαίνει, καθως το PC2 δεν έχει κανόνα δρομολόγησης για την διεύθυνση 172.17.17.1.

[1] Ο αντίστοιχος κανόνας στην αντίστροφη διαδρομή είναι ο R1 → PC2, ο οποίος προστέθηκε χειροκίνητα στο ερώτημα 3.7

3.16 Στο PC2:

```
route delete 192.168.1.0/24
```

3.17 Στο PC2:

```
route add default 192.168.2.1
```

3.18 Επιτυγχάνει.

3.19 Στο 1ο ping, το PC2 δεν είχε κανόνα δρομολόγησης για την διεύθυνση 172.17.17.1 οπότε η αποστολή πακέτων απέτυχε με αυτό το μήνυμα λάθους.

Στο 2ο ping, το Echo Request ακολούθησε την διαδρομή PC2 → R2 → R1 και το Echo Reply που εξέδωσε ο R1 ακολούθησε την διαδρομή R1 → R2 → PC2. (ο R1 έχει κανόνας δρομολόγησης για την διεύθυνση του PC2 από προηγουμένως)

Άσκηση 4

4.1 Στο PC3:

```
ifconfig em0 192.168.2.3/24
route add default 192.168.2.1
```

4.2 Στον R1:

```
sysrc ifconfig_em2="172.17.17.5/30"
service netif restart
```

4.3 Στον R2:

```
sysrc ifconfig_em2="172.17.17.9/30"
service netif restart
```

4.4 Στον R3:

```
sysrc ifconfig_em0="172.17.17.6/30"
sysrc ifconfig_em1="172.17.17.10/30"
service netif restart
```

4.5 Στον R3:

```
sysrc gateway_enable="YES"
service routing restart
```

4.6 Στον R1:

```
route add 192.168.2.0/24 172.17.17.2
```

4.7 Στον R2:

```
route add 192.168.1.0/24 172.17.17.1
```

4.8 Στον R3:

```
route add 192.168.1.0/24 172.17.17.5
route add 192.168.2.0/24 172.17.17.9
```

4.9 Στον R1:

```
route add 192.168.2.3 172.17.17.6
```

Το ότι πρόκειται για διαδρομή προς υπολογιστή φαίνεται από την σημαία `H` στον πίνακα δρομολόγησης.

4.10 3 (PC1 → R1, R1 → R2, R2 → PC2)

4.11 Το TTL έχει μειωθεί 2 φορές. Όμως το TTL μειώνεται μια φορά σε κάθε δρομολογητή. Άρα πρόκειται για 3 βήματα.

4.12 4 (PC1 → R1, R1 → R3, R3 → R2, R2 → PC3)

4.13 Το TTL έχει μειωθεί 2 φορές. Όμως το TTL μειώνεται μια φορά σε κάθε δρομολογητή. Άρα πρόκειται για 3 βήματα.

4.14 PC1 → R1 → R3 → R2 → PC3

4.15 PC3 → R2 → R1 → PC1

Το PC3 για την διεύθυνση 192.168.1.2 του 192.168.1.0/24 προωθεί προς τον R2.

Ο R2 για την διεύθυνση 192.168.1.2 του 192.168.1.0/24 προωθεί προς τον R1.

Ο R1 για την διεύθυνση 192.168.1.2 του 192.168.1.0/24 προωθεί απευθείας προς το LAN1, οπότε το πακέτο φθάνει στο PC1.

4.16 Στον R2:

```
tcpdump -i em1 -n
```

4.17 Όχι δεν φθάνουν.

4.18 Τα UDP πακέτα φθάνουν στο PC3. Το PC3 απαντά με ICMP Port Unreachable, το οποίο προωθεί προς τον R2, που το προωθεί προς τον R1. Ωστόσο, παρόλο που ο R2 προωθεί προς τον R1, τα πακέτα δεν φθάνουν ποτέ στον R1 διότι η σύνδεση R1-WAN1 δεν λειτουργεί, οπότε το πακέτο χάνεται και δεν φθάνει ποτέ στο PC1 για να το έδειχνε το traceroute.

4.19 Ναι.

4.20 Στον R1:

```
route change 192.168.2.0/24 172.17.17.6
```

Στον R2:

```
route change 192.168.1.0/24 172.17.17.10
```

Το traceroute από PC1 προς PC2 και προς PC3 επιτυγχάνει και δείχνει διαδρομή μέσω R3.

4.21 Στον R1:

```
route get 192.168.2.2
route get 192.168.2.3
```

Παρατηρούμε πως πρόκειται για διαφορετικές εγγραφές, όπως ακριβώς τις εισάγαγαμε χωριστά. Πιο συγκεκριμένα, για το PC2 χρησιμοποιείται η εγγραφή για το δίκτυο 192.168.2.0/24, ενω για το PC3 η χωριστή εγγραφή για τον host 192.168.2.3 (έχει και την σημαία `H`, οπότε αντιλαμβάνομαστε πως πρόκειται για χωριστή εγγραφή που αφορά έναν και μόνο host)

4.22 Αυτή που αφορά στο 192.168.2.3 ως χωριστό host καθώς έχει μεγαλύτερο πρόθεμα δικτύου και συγκεκριμένα 32 bits (που ειναι και το μέγιστο), ενώ η άλλη εγγραφή έχει πρόθεμα δικτύου 24 bits.

4.23 Στον R3:

```
route change 192.168.2.0/24 172.17.17.5
```

4.24 Όχι αποτυγχάνει και, μάλιστα, παραλαμβάνεται το μήνυμα λάθους ICMP Time exceeded από την IP διεύθυνση 172.17.17.6, δηλαδή από τον R3.

4.25 Το PC1 εκδίδει ένα IP πακέτο (το Echo Request) με παραλήπτη το PC2, και το προωθεί στον R1. Ο R1 το προωθεί στον R3, που με την σειρά του το προωθεί στον R3, κ.ο.κ. Ο κύκλος αυτός τερματίζει διότι σε κάθε βήμα το πεδίο TTL μειώνεται κατά 1, και έτσι, κάποια στιγμή μηδενίζεται, οπότε ο τελευταίος δρομολογητής, όπου γίνεται TTL = 0, απορρίπτει το πακέτο, στέλνοντας ένα ICMP Time exceeded προς τον αποστολέα, δηλαδή προς τον R1. Το πεδίο TTL έχει αρχική τιμή 64, οπότε μηδενίζεται στον 64ο δρομολόγητη από όπου θα περάσει, και επειδή $64 \equiv 0 \mod 2$, αυτός ο δρομολογητής είναι ο R3, όπως πράγματι είδαμε και να συμβαίνει (αυτός έστειλε στο PC1 το ICMP Time exceeded).

4.26 Στον R3:

```
tcpdump -i em0 -e -n 'icmp[icmptype]=icmp-echo
```

4.27 63

4.28 Ο R1 εκπεμπεί πακέτα με περιττό TTL (ξεκινώντας από το 63) και ο R3 με άρτιο TTL (ξεκινώντας από το 62), οπότε ο R1 παράγει 32 πακέτα και ο R3 παράγει 31 πακέτα.

Πράγματι, τρέχουμε στον R3 δύο χωριστά tcpdump που να φιλτράρουν και βάση του περιορισμού:

```
tcpdump -i em0 -e -n 'icmp[icmptype]=icmp-echo <MAC-R1-WAN2>
tcpdump -i em0 -e -n 'icmp[icmptype]=icmp-echo <MAC-R3-WAN2>
```

Παρατηρούμε πως πράγματι ο R1 παράγει 32 πακέτα και ο R3 παράγει 31 πακέτα.

4.29 Στον R1:

```
tcpdump -i em0 -e -n 'icmp[icmptype]=icmp-echo
```

Στον R3:

```
tcpdump -i em0 -e -n 'icmp[icmptype]=icmp-timxceed
```

4.30 Εμφανίζονται 64 βήματα και είναι η διαδρομή PC1 → R1 → R3 → R1 ··· R3.

4.31 Το PC1 απέστειλε 64 πακέτα Echo Request (με τιμές TTL=1 ως 64).

Στο WAN1 προωθούνται τα πακέτα με TTL >= 2 και καθένα εμφανίζεται στο WAN1 TTL-1 φορές συνολικά. Συνεπώς συνολικά στο WAN1 εμφανίζονται:

$$(2-1)+(3-1)+\cdots+(64-1)=1+2+\cdots+63=\frac{63\cdot 64}{2}=2016\text{πακέτα Echo Requests}$$

Πράγματι, και η καταγράφη έδειξε πως στο WAN2 στάλθηκαν συνολικά 2016 Echo Requests.

4.32 32. Στο WAN2 μηνύματα ICMP Time Exceeded εκδίδει μόνο ο R3 (ο R1 το προωθεί απευθείας στο PC1). Ο R3 είναι ο τελικός δρομολογητής (όπου το TTL μηδενίζεται) για αρχικά άρτια TTL >=2. Echo Requests εκπέμπονται σε όλες τις τιμές $2,\cdots,64$, οπότε τελικά στο WAN2 εκπέμπονται 32 μηνύματα ICMP Time Exceeded.

Πράγματι, και η καταγραφή έδειξε πως στο WAN2 στάλθηκαν συνολικά 32 μηνύματα ICMP Time Exceeded.

Άσκηση 5

5.1 Το LAN1 πρέπει να περιέχει 120 + 2 = 122 IP διευθύνσεις (+1 διεύθυνση broadcast, +1 την μηδενική διεύθυνση). Απαιτούνται $\lceil \log_2 122 \rceil = 7$ bits για αυτό, οπότε απαιτείται πρόθεμα δικτύου το πολύ 25 bits.

Αναθέτουμε στο LAN1 το 172.17.17.0/25.

5.2 Με όμοιο σκεπτικό, το LAN2 απαιτεί $\lceil \log_2 (60+2) \rceil = 6$ bits, οπότε του αναθέτουμε το 172.17.17.192/26.

5.3 Το LAN3 απαιτεί $\lceil \log_2 (30+2) \rceil = 5$ bits, οπότε του αναθέτουμε το 172.17.17.160/27.

5.4 Στον R1:

```
sysrc ifconfig_em1="172.17.17.129/30"
sysrc ifconfig_em2="172.17.17.133/30"
```

5.5 Στον R1:

```
sysrc ifconfig_em0="172.17.17.127/25"
```

Στο PC1:

```
sysrc ifconfig_em0="172.17.17.1/25"
```

5.6 Στον R2:

```
sysrc ifconfig_em0="172.17.17.130/30"
sysrc ifconfig_em2="172.17.17.138/30"
```

5.7 Στον R2:

```
sysrc ifconfig_em1="172.17.17.193/26"
```

Στο PC2:

```
sysrc ifconfig_em0="172.17.17.253/26"
```

Στο PC3:

```
sysrc ifconfig_em0="172.17.17.254/26"
```

5.8 Στον R3:

```
sysrc ifconfig_em0="172.17.17.134/30"
sysrc ifconfig_em1="172.17.17.137/30"
```

5.9 Στον R2:

```
sysrc ifconfig_em2="172.17.17.190/26"
```

Στο PC4:

```
sysrc ifconfig_em0="172.17.17.161/26"
```

5.10 Στο PC1:

```
route add default 172.17.17.127
```

Στο PC2:

```
route add default 172.17.17.193
```

Στο PC3:

```
route add default 172.17.17.193
```

Στο PC4:

```
route add default 172.17.17.190
```

5.11 Στον R1:

```
route add 172.17.17.192/26 172.17.17.130
route add 172.17.17.160/27 172.17.17.130
```

5.12 Στον R2:

```
route add 172.17.17.0/25 172.17.17.137
route add 172.17.17.160/27 172.17.17.137
```

5.13 Στον R3:

```
route add 172.17.17.0/25 172.17.17.133
route add 172.17.17.193/27 172.17.17.133
```

5.16 Η επικοινωνία λειτουργεί.

Άσκηση 6

6.1 PC2: 08:00:27:9a:d7:2e

PC3: 08:00:27:13:58:da

6.2 Στο PC2:

```
ifconfig em0 172.17.17.254/26
```

6.3 Ναι ο πυρήνας του FreeBSD μας ενημερώνει απευθείας ότι το μηχάνημα με την MAC διεύθυνση του PC3 χρησιμοποιεί την ίδια IP διεύθυνση με αυτόν.

6.4 Ναι, εκεί μας ενημερώνει αντίστοιχα ότι το μηχάνημα με την MAC διεύθυνση του PC2 χρησμοποιεί την ίδια διεύθυνση με αυτόν.

6.5 Ναι έχει.

Το νόημα του μηνύματος λάθους είναι να ενημερώσει άμεσα τον διαχειριστή του συστήματος, πως δεύτερο μηχάνημα έχει την ίδια IP διεύθυνση με αυτό, οπότε πρέπει να την αλλάξει, διότι δεν θα είναι σαφές σε ποιο μηχάνημα πρέπει να αποστέλλονται τα πακέτα.

6.6 Όχι. Αφού επαναρυθμίσαμε την IP διεύθυνση της διεπαφής `em0`, όλες οι εγγραφές που τελικά θα οδηγούσαν σε αποστολή πακέτω στην `em0` διαγράφηκαν.

6.7 Στο PC2:

```
route add default 182.17.17.193
```

6.8 Στα PC2, PC3, R2:

```
arp -a -d
```

6.9 Στον R2:

```
tcpdump -i em1 -e -n 'arp'
```

6.10 Στα PC2, PC3:

```
tcpdump -i em0 -e -n 'tcp'
```

6.11 Απέτυχε και εμφανίστηκε ένα μήνυμα λάθους Connection reset by peer, που είναι σφάλμα του TCP και συγκεκριμένα ότι ο άλλος host εξέδωσε ένα RST.

6.12 Ήταν επιτυχής.

6.13 Έχει καταγράφει η κοινή τους IPv4 διεύθυνση 172.17.17.254 στην MAC διεύθυνση 08:00:27:13:58:da, που ανήκει στο PC3.

6.14 Πρώτα το PC2 και μετά το PC3.

6.15 Στο PC3.

6.16 Στο PC3. (το καταλάβαμε διότι έχω ορίσει hostname PC3 στο PC3, οπότε μόλις συνδεθήκαμε εμφανίστηκε το hostname στο prompt του shell)

6.17 Μπορούμε με τους εξής τρόπους:

- Από τις καταγραφές στα PC2, PC3, όπου παρατηρούμε ότι πακέτα φθάνουν στο PC3 αλλά όχι στο PC2.
- Τρέχοντας `ifconfig` στην κονσόλα του μηχανήματος που συνδεθήκαμε και βλέποντας την MAC διεύθυνση.

6.18 Προκειμένου να κατανοήσουμε καλύτερα την χρονική αλληλουχία των πακέτων, αφού καθαρίσαμε τους πίνακες ARP εκ νεόυ, κάναμε μια καταγραφή στον R2 για όλα τα πακέτα και επίσης μια καταγραφή στο PC1 για TCP πακέτα.

Παρατηρούμε πως αφού ο R2 εκδόσει ARP Request για την διεύθυνση 172.17.17.254, απαντά πρώτα το PC2 την διεύθυνση MAC του. Άμεσα (45 μs μετά) ο R2 προωθεί στο PC2 το 1ο πακέτο που στέλνει το PC2, που είναι το τεμάχιο SYN που είχε στείλει το PC1. Λίγο αργότερα (313 μs μετά) φθάνει στον R2 το ARP Reply του PC3. Έτσι ο R2 πια αποστέλλει τα πακέτα με προορισμό την 172.17.17.254 αντί προς το PC2, πλέον προς το PC3. Ωστόσο, το τεμάχιο SYN έχει ήδη παραδοθεί στο PC2, που άμεσα απαντά με τεμάχιο SYNACK, το οποίο προωθείται στον R2, που με την σειρά του το δρομολογεί προς το PC1. Μόλις το τεμάχιο SYNACK φθάσει στο στο PC1, το PC1 εκδίδει το 3ο και τελευταίο τεμάχιο της τριπλής χειαραψίας, που είναι το ACK για το SYN της απέναντι του πλευράς. Αυτό όμως, το τεμάχιο, ο R2 αυτή τη φορά το παραδίδει στο PC3. Έτσι, το PC3 παραλαμβάνει ένα πακέτο ACK χωρίς να έχει εγκαθιδρυθεί TCP σύνδεση με αυτό, γεγονός που δεν είναι αποδεκτό από το πρωτόκολλο TCP, και υποδηλώνει ότι μάλλον κάποιο λάθος έχει συμβεί. Έτσι, το PC3 εκδίδει ένα τεμάχιο RST με παραλήπτη το PC1. Έτσι, το PC1 εγκαταλείπει την σύνδεση με μήνυμα λάθους προς τον χρήστη.

Ωστόσο, το PC1, είχε στείλει μαζί με το ACK για το SYNACK που ελάβε, και το πρώτο τεμάχιο δεδομένων (γίνεται στο TCP για να μειωθούν τα latencies, και ούτως ή άλλως στην χειρότερη περίπτωση απλά τα δεδομένα δεν θα επιβεβαιωθούν ότι παρελήφθησαν) Έτσι, πριν προλάβει να φθάσει το RST στο PC1 έχει εκπεμπφθεί τεμάχιο δεδομένων, που και αυτό φθάνει στο PC3, που φυσικά, απαντά ξανά με RST, οπότε φθάνει και δευτέρο RST στο PC1.

Ταυτόχρονα, το PC2 δεν έχει ιδέα για όσα έχουν συμβεί, έχει στείλει ένα SYNACK και περιμένει να του έρθει ένα ACK. Επειδή δεν του έρχεται, εκπνέει το χρονόμετρο, και έτσι πρακτικά το PC2 θεωρεί πως είτε το SYNACK του είτε το ACK του PC2 χάθηκαν, οπότε επανεκδίδει το τεμάχιο SYNACK. Πλέον το PC1 απαντά με RST καθώς η σύνδεση έχει εγκαταλειφθεί. Δυστυχώς, όμως, το RST προωθείται από τον R2 προς το PC3, που το αγνοεί, και έτσι το PC2 δεν μαθαίνει για το τι έχει σύμβει. Αυτές οι επαναποστολές, λόγω του exponential backoff στον χρόνο αναμετάδοσης, επαναλαμβάνονται ανά εκθετικά αυξανόμενα διαστήματα μέχρι που το PC2 εγκαταλείπει.

Την **2η φορά που εκτελούμε ssh** από την αρχή το πακέτα SYN και όλα τα επόμενα προωθούνται από τον R2 στο PC3. Έτσι η σύνδεση εγκαθιδρύεται σωστά, όλα τα τεμάχια προς την 172.17.17.254 πρωοθούνται προς το PC3, και έτσι η ssh σύνδεση λειτουργεί φυσιολογικά.

6.19 Εξήγηθηκε αναλυτικά ακριβώς από πάνω. (ερώτημα 6.18)