



ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4: ΠΡΩΤΟΚΟΛΛΟ IPV4 ΚΑΙ ΘΡΥΜΜΑΤΙΣΜΟΣ



1 ΝΟΕΜΒΡΙΟΥ, 2022

ΘΟΔΩΡΗΣ ΑΡΑΠΗΣ – EL18028

Όνοματεπώνυμο: Θεodorής Αράπης		Ομάδα: 2
Όνομα PC/ΛΣ: DESKTOP-JGHL94V/ WINDOWS 10		Ημερομηνία: 1/11/2022
Διεύθυνση IP: 192.168.1.6 (1-2), 192.168.1.4 (3-4)		Διεύθυνση MAC: 70-85-C2-88-FD-B1

Άσκηση 1: Μετρήστε την καθυστέρηση

1.1

Η ακριβής σύνταξη της εντολής είναι «ping -4 -n 3 www.mit.edu».

```
C:\WINDOWS\system32>ping -4 -n 3 www.mit.edu
```

Το «-4» δηλώνει πως θέλουμε να επιβληθεί το πρωτόκολλο IPv4 και το «-n 3» δηλώνει ότι θέλουμε να σταλούν 3 πακέτα.

1.2

Το φίλτρο σύλληψης «not broadcast and not multicast» δηλώνει πως θέλουμε να αποφύγουμε την καταγραφή πακέτων που γίνονται broadcast καθώς και πακέτων που γίνονται multicast. Με άλλα λόγια συλλαμβάνουμε μόνο unicast πακέτα περιορίζοντας τον θόρυβο πακέτων broadcast και multicast που λαμβάνει ο υπολογιστής μας.

1.3

Όπως φαίνεται και από το screenshot, έχουμε 0% ποσοστό απωλειών (0% loss), ενώ η μέση καθυστέρηση (Average) είναι 85ms.

```
C:\WINDOWS\system32>ping -4 -n 3 www.mit.edu

Pinging e9566.dscb.akamaiedge.net [95.101.209.207] with 32 bytes of data:
Reply from 95.101.209.207: bytes=32 time=85ms TTL=54
Reply from 95.101.209.207: bytes=32 time=85ms TTL=54
Reply from 95.101.209.207: bytes=32 time=85ms TTL=54

Ping statistics for 95.101.209.207:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 85ms, Maximum = 85ms, Average = 85ms
```

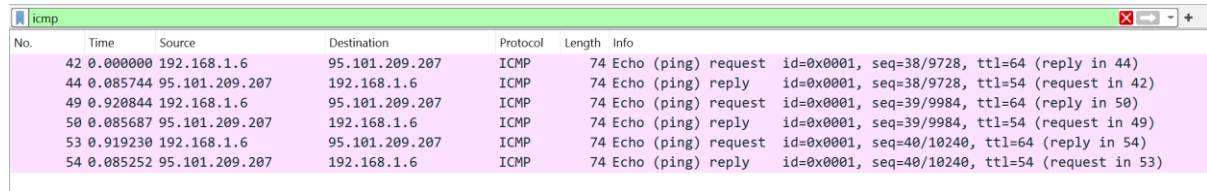
1.4

Όπως φαίνεται και από το screenshot παραπάνω, οι τιμές του RTT (Round Trip Times) είναι:

Minimum = 85ms, Maximum = 85ms, Average = 85ms

1.5

Από το ακόλουθο Screenshot βλέπουμε ότι πράγματι το minimum, το maximum και επομένως το average RTT είναι 85ms (Echo (ping) request). Οπότε οι τιμές συμφωνούν.



No.	Time	Source	Destination	Protocol	Length	Info
42	0.000000	192.168.1.6	95.101.209.207	ICMP	74	Echo (ping) request id=0x0001, seq=38/9728, ttl=64 (reply in 44)
44	0.085744	95.101.209.207	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=38/9728, ttl=54 (request in 42)
49	0.920844	192.168.1.6	95.101.209.207	ICMP	74	Echo (ping) request id=0x0001, seq=39/9984, ttl=64 (reply in 50)
50	0.085687	95.101.209.207	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=39/9984, ttl=54 (request in 49)
53	0.919230	192.168.1.6	95.101.209.207	ICMP	74	Echo (ping) request id=0x0001, seq=40/10240, ttl=64 (reply in 54)
54	0.085252	95.101.209.207	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=40/10240, ttl=54 (request in 53)

1.6

Αρκεί να εφαρμόσουμε το φίλτρο «ip» ώστε να μας εμφανιστούν μόνο τα πακέτα IPv4.

1.7

Αρκεί να εφαρμόσουμε το φίλτρο «icmp» ώστε να μας εμφανιστούν μόνο τα πακέτα που διέπονται από το πρωτόκολλο ICMP.

1.8

Εξετάζουμε το πεδίο Type της επικεφαλίδας ICMP την πλαίσίων με Source IP την δική μας και βλέπουμε ότι έχουν σταλθεί 3 πακέτα ICMP τύπου Echo (ping) request (8).

1.9

Η διεύθυνση πηγής είναι: 192.168.1.6, ενώ η διεύθυνση προορισμού είναι: 95.101.209.207 (www.mit.edu).

1.10

Εξετάζουμε το πεδίο Type της επικεφαλίδας ICMP την πλαίσίων με Destination IP την δική μας και βλέπουμε ότι έχουν ληφθεί 3 πακέτα ICMP τύπου Echo (ping) reply (0).

1.11

Η διεύθυνση πηγής είναι: 95.101.209.207 (www.mit.edu), ενώ η διεύθυνση προορισμού είναι: 192.168.1.6.

1.12

Παρατηρούμε ότι έχει αλλάξει η IPv4 διεύθυνση του host www.mit.edu από 18.7.22.83 σε 95.101.209.207, η τιμή TTL (Time To Live) είναι πλέον 54 ενώ τότε ήταν 242, οι Τιμές RTT στην δική μας περίπτωση είναι 85 ms ενώ τότε ήταν 136 ms και τέλος, εμείς στείλαμε 3 πακέτα ενώ στην παλιά καταγραφή είχαν σταλθεί 4,

Άσκηση 2: Περισσότερα για το ping

2.1

Χρησιμοποιήσαμε 3 φορές την εντολή ping, για τις 3 διαφορετικές περιπτώσεις που είχαμε. Η σύνταξή τους ήταν: «ping -4 -n 5 *hostname*», όπου *hostname* είναι η διεύθυνση IPv4 που θέλουμε να σταλθεί το αίτημα Echo. Συγκεκριμένα:

i) 192.168.1.5 (Άλλο PC συνδεδεμένο στο δίκτυό μας)

ii) 192.168.1.6 (IPv4 της διεπαφής δικτύου μας)

iii) 127.0.0.1 (loopback)

Θα μπορούσαμε να χρησιμοποιήσουμε και μία εντολή για όλα τα αιτήματα, η οποία θα είχε την εξής σύνταξη:

«ping -4 -n 5 192.168.1.6 && ping -4 -n 5 192.168.1.6 && ping -4 -n 5 127.0.0.1»

```
C:\WINDOWS\system32>ping -4 -n 5 192.168.1.6 && ping -4 -n 5 192.168.1.6 && ping -4 -n 5 127.0.0.1

Pinging 192.168.1.6 with 32 bytes of data:
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.6:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

Pinging 192.168.1.6 with 32 bytes of data:
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64
Reply from 192.168.1.6: bytes=32 time<1ms TTL=64

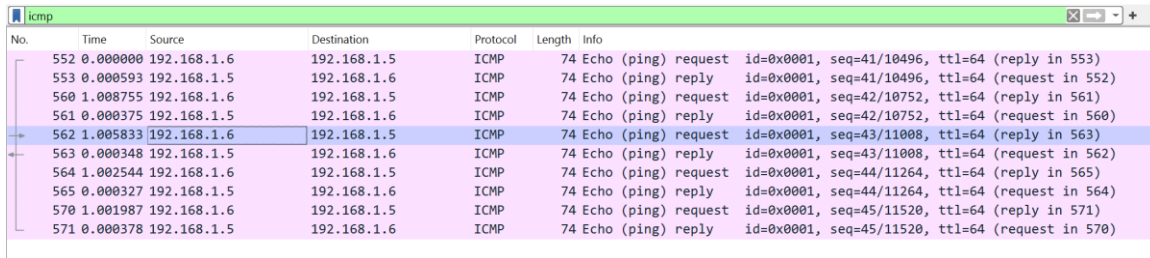
Ping statistics for 192.168.1.6:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=64
Reply from 127.0.0.1: bytes=32 time<1ms TTL=64
Reply from 127.0.0.1: bytes=32 time<1ms TTL=64
Reply from 127.0.0.1: bytes=32 time<1ms TTL=64
Reply from 127.0.0.1: bytes=32 time<1ms TTL=64

Ping statistics for 127.0.0.1:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

2.2

Το Wireshark έχει καταγράψει μόνο 5 από τα 15 αιτήματα που στείλαμε, μόνο αυτά που στάλθηκαν προς άλλη συσκευή στο δίκτυό μας.



No.	Time	Source	Destination	Protocol	Length	Info
552	0.000000	192.168.1.6	192.168.1.5	ICMP	74	Echo (ping) request id=0x0001, seq=41/10496, ttl=64 (reply in 553)
553	0.000593	192.168.1.5	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=41/10496, ttl=64 (request in 552)
560	1.000755	192.168.1.6	192.168.1.5	ICMP	74	Echo (ping) request id=0x0001, seq=42/10752, ttl=64 (reply in 561)
561	0.000375	192.168.1.5	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=42/10752, ttl=64 (request in 560)
562	1.005833	192.168.1.6	192.168.1.5	ICMP	74	Echo (ping) request id=0x0001, seq=43/11008, ttl=64 (reply in 563)
563	0.000348	192.168.1.5	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=43/11008, ttl=64 (request in 562)
564	1.002544	192.168.1.6	192.168.1.5	ICMP	74	Echo (ping) request id=0x0001, seq=44/11264, ttl=64 (reply in 565)
565	0.000327	192.168.1.5	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=44/11264, ttl=64 (request in 564)
570	1.001987	192.168.1.6	192.168.1.5	ICMP	74	Echo (ping) request id=0x0001, seq=45/11520, ttl=64 (reply in 571)
571	0.000378	192.168.1.5	192.168.1.6	ICMP	74	Echo (ping) reply id=0x0001, seq=45/11520, ttl=64 (request in 570)

2.3

Ο προορισμός τους ήταν η άλλη συσκευή που είναι συνδεδεμένη στο δίκτυό μας, δηλαδή η συσκευή με IPv4: 192.168.1.5

2.4

Όχι, δεν παρατηρήσαμε αποστολή μηνυμάτων ICMP request στο δίκτυο με πηγή και προορισμό την IPv4 του υπολογιστή μας. Αυτό συμβαίνει διότι, σύμφωνα με το σχήμα, εφόσον η IP μας είναι τοπική διεύθυνση IPv4, τοποθετείται στην ουρά εισόδου IPv4, με αποτέλεσμα να μην φεύγει ποτέ προς το τοπικό δίκτυο.

2.5

Όχι, δε παρατηρούμε αποστολή μηνυμάτων ICMP Echo Request προς τη διεύθυνση του βρόχου επιστροφής (127.0.0.1), ακριβώς επειδή, όπως και πριν, η διεύθυνση αυτή είναι τοπική, συνεπώς τα πακέτα δε προωθούνται ποτέ στο τοπικό δίκτυο και επομένως δε γίνονται Capture.

2.6

Ένας λόγος για να κάνουμε ping στην διεύθυνσή μας είναι για να εξετάσουμε ότι η κάρτα δικτύου λειτουργεί κανονικά. Αυτό γίνεται εφόσον είμαστε συνδεδεμένοι στο διαδίκτυο. Από την άλλη, κάνουμε ping στη διεύθυνση βρόχου επιστροφής όταν θέλουμε να επιβεβαιώσουμε ότι το TCP/IP software έχει εγκατασταθεί, έχει ξεκινήσει και λειτουργεί κανονικά, κάτι που δεν απαιτεί συνδεσιμότητα στο διαδίκτυο. Οπότε, εάν θέλουμε να κάνουμε troubleshooting, ξεκινάμε κάνοντας Ping σε αυτή τη διεύθυνση loopback, ύστερα στην IPv4 μας εφόσον λάβαμε απάντηση προηγουμένως, στη συνέχεια στο router μας και τέλος σε μια IP διεύθυνση εξωτερικού δικτύου, ώστε να σιγουρευτούμε ότι όλα πάνε καλά.

2.7

Κάνουμε ping προς τις δύο διευθύνσεις και έχουμε τα εξής αποτελέσματα:

```
C:\WINDOWS\system32>ping www.netflix.com

Pinging apiproxy-website-nlb-prod-3-ac110f6ae472b85a.elb.eu-west-1.amazonaws.com [2a05:d018:76c:b684:8ab7:ac02:667b:e863] with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 2a05:d018:76c:b684:8ab7:ac02:667b:e863:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\WINDOWS\system32>ping www.amazon.com

Pinging e15316.dsca.akamaiedge.net [2a02:26f0:c000:285::3bd4] with 32 bytes of data:
Reply from 2a02:26f0:c000:285::3bd4: time=8ms
Reply from 2a02:26f0:c000:285::3bd4: time=9ms
Reply from 2a02:26f0:c000:285::3bd4: time=9ms
Reply from 2a02:26f0:c000:285::3bd4: time=8ms

Ping statistics for 2a02:26f0:c000:285::3bd4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 8ms, Maximum = 9ms, Average = 8ms

C:\WINDOWS\system32>
```

Ενώ το και οι δύο σελίδες φορτώνουν κανονικά στον browser, παρατηρούμε ότι η σελίδα Netflix δεν ανταποκρίνεται στα Echo request που της στέλνουμε. Δύο λόγοι που μπορεί να συμβαίνει αυτό είναι:

- Μπορεί κατά τη διαδρομή των πακέτων από τον υπολογιστή μας στον server του Netflix να παρεμβάλλεται κάποιο τοίχος προστασίας (firewall), το οποίο μπλοκάρει τα μηνύματα πρωτοκόλλου ICMP, ενδεχομένως για να μη προκληθεί συμφόρηση των server από τέτοια, με αποτέλεσμα να χάνονται πακέτα κατά το request.
- Υπάρχει περίπτωση ο κόμβος προορισμού ή κάποια ενδιαμέση συσκευή να μην είναι επαρκώς πληροφορημένη για το δίκτυο του αποστολέα, δηλαδή του υπολογιστή μας και έτσι να μην είναι δυνατή η σωστή επιστροφή της απάντησης και ως εκ τούτου να χάνονται πακέτα κατά το reply.

Άσκηση 3: Επικεφαλίδες IPv4

3.1

Το φίλτρο σύλληψης που χρησιμοποιήσαμε είναι το «host 147.102.40.15».

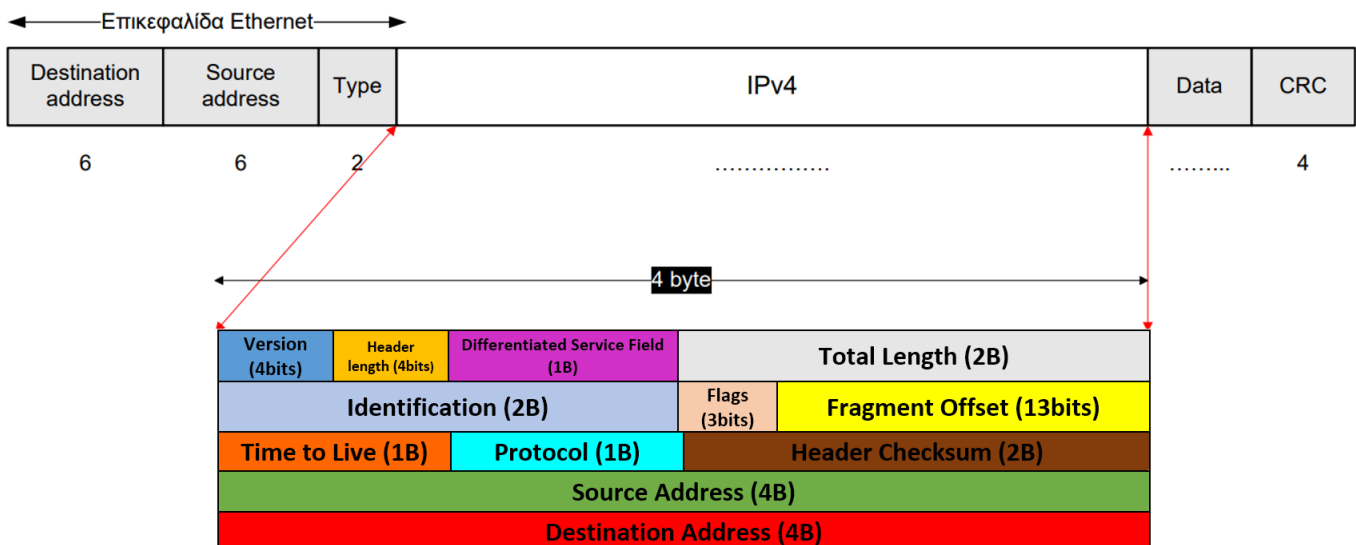
3.2

Το φίλτρο απεικόνισης είναι το «ip.src == 192.168.1.4», δηλαδή ip διεύθυνση αποστολέα είναι η ip της κάρτας δικτύου μας.

3.3

Τα πεδία της επικεφαλίδας IPv4 είναι:

```
Internet Protocol Version 4, Src: 192.168.1.4, Dst: 147.102.40.15
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 48
    Identification: 0xa457 (42071)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x194f [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.4
    Destination Address: 147.102.40.15
```



3.4

Ελέγχοντας όλα τα πλαίσια, παρατηρούμε ότι τα πεδία που αλλάζουν τιμές ανά πλαίσιο είναι: το Total Length, το Identification και το Header Checksum.

3.5

Ναι, το μήκος της επικεφαλίδας IPv4 είναι ίδιο σε όλα τα πλαίσια και ίσο με 20 Bytes.

3.6

Ελάχιστο μήκος πακέτου 40 bytes, ενώ μέγιστο μήκος πακέτου 58 bytes.

3.7

Το πεδίο Differentiated Services Field έχει τιμή 0x00 σε όλα τα πακέτα, το οποίο αντιστοιχεί στο Standard Service Class με Default Forwarding συμπεριφορά. Όταν κάποιο traffic δεν πληροί προϋποθέσεις κάποιας άλλης κλάσης, τότε χρησιμοποιεί τη Default Forwarding (best effort forwarding χαρακτηριστικά), η οποία, όμως, δεν εξασφαλίζει ότι τα δεδομένα θα διαδοθούν ή ότι θα διαδοθούν αξιόπιστα.

3.8

Για το πεδίο Identification παρατηρούμε ότι οι τιμές των πακέτων αυξάνονται κατά 1 καθώς ελέγχουμε τα πακέτα από πάνω προς τα κάτω. Το πρώτο πακέτο έχει τιμή 0xa454 και το τελευταίο 0xa47e (Δεν έχουμε πακέτα που να έχουν σπάσει σε Fragments οπότε δεν συναντάμε κοινά Identification).

3.9

Η σημαία Don't fragment έχει τιμή 1 (Set), δηλαδή να μην σπάσει σε fragments το πακέτο.

3.10

Το πεδίο Fragment Offset έχει τιμή 0 σε όλα τα πακέτα (αφού κανένα δεν έχει σπάσει σε fragments άρα έχουν μηδενικό position)

3.11

Το πεδίο protocol έχει τιμή 0x06 και ανήκει στο πρωτόκολλο TCP.

3.12

Το header checksum υπολογίζεται ως εξής:

Χωρίζουμε το IP header σε 16bit λέξεις και αφού τις αθροίσουμε, παίρνουμε το συμπλήρωμα ως προς ένα του αθροίσματος (με κάποια έξτρα βήματα εάν προκύψει κρατούμενο

και έχοντας θεωρήσει ως 0 την τιμή του πεδίου header checksum). Συνεπώς, είναι προφανές πως το header checksum βασίζεται στις τιμές των πεδίων της IP επικεφαλίδας του εκάστοτε πακέτου, κάποιες από τις οποίες όπως είδαμε διαφέρουν (Total Length εν γένει, αλλά όχι απαραίτητα διαφορετικό, αλλά Identification μοναδικό σε κάθε πακέτο) δίνοντας έτσι μοναδική τιμή header checksum σε κάθε πακέτο.

Άσκηση 4: Θρυμματισμός (Fragmentation) στο IPv4

4.1

Η εντολή ping θα έχει σύνταξη «ping -4 -n 1 -l size -f ip_address», όπου:

- **-4** δηλώνει ότι επιβάλλεται IPv4 πρωτόκολλο
- **-n 1** δηλώνει ότι θέλουμε να στείλουμε ένα πακέτο μόνο
- **-l size** δηλώνει το μέγεθος του πακέτου που στέλνουμε
- **-f** δηλώνει ότι δεν θέλουμε να γίνει θρυμματισμός

4.2

Αρχικά στέλνουμε πακέτο echo request προς την κινητή συσκευή μας με IP διεύθυνση 192.168.1.7 και μέγεθος πακέτου 1480 bytes.

```
C:\WINDOWS\system32>ping -4 -n 1 -l 1480 -f 192.168.1.7

Pinging 192.168.1.7 with 1480 bytes of data:
Packet needs to be fragmented but DF set.

Ping statistics for 192.168.1.7:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

Βλέπουμε ότι επιχειρήθηκε από τον router μας να θρυμματιστεί το πακέτο αλλά αφού είχαμε ενεργοποιήσει το flag -f, το πακέτο δεν θρυμματίστηκε. Μειώνοντας κατά 1 byte το μέγεθος πακέτου που στέλνουμε σε κάθε προσπάθεια, καταλήγουμε ότι το μέγιστο μέγεθος πακέτου που μπορούμε να στείλουμε είναι 1472 bytes.

4.3

Σύμφωνα με τα παραπάνω, η μικρότερη τιμή για την οποία απαιτείται θρυμματισμός είναι 1473 bytes.

4.4

Χρησιμοποιήσαμε το φίλτρο σύλληψης «not broadcast and not multicast».

4.5

Χρησιμοποιήσαμε το φίλτρο απεικόνισης «ip.addr == 192.168.1.7».

4.6

Όχι δεν παράγονται αφού το μέγεθος του πακέτου είναι τόσο μεγάλο που απαιτείται fragmentation, αλλά εφόσον έχουμε βάλει το flag don't fragment τότε δεν μπορεί να θρυμματιστεί το πακέτο και άρα δεν γίνεται να σταλθεί.

4.7

Το Wireshark κατέγραψε ότι το μέγεθος του Ethernet frame που στάλθηκε όταν κάναμε το echo request ήταν 1514 bytes. Από αυτά, τα 14 bytes αποτελούν το Ethernet Header, οπότε το MTU έχει μέγεθος 1500 bytes (Το FCS δεν καταγράφεται από το Wireshark).

4.8

Χωρίς Fragmentation, το ICMP πρέπει να ναι 1472 bytes για IPv4 πακέτα μέγιστου μήκους. Όταν έχουμε Fragmentation, τότε το μέγιστο ICMP μπορεί να φτάσει $65.535 - 20 - 8 = 65.507$ bytes, όπου 20 bytes το IP header και 8 bytes το ICMP header. Ωστόσο, το λειτουργικό μας σύστημα (Windows) επιτρέπει έως και 65.500 bytes, σε αντίθεση με ένα σύστημα Linux που επιτρέπει έως και 65.507.

4.9

Για 65.507 bytes ICMP πακέτου, η εντολή αποτυγχάνει. Η μέγιστη τιμή, για την οποία πετυχαίνει είναι 65.500 bytes.

4.10

Το μέγεθος του μεγαλύτερου IPv4 πακέτου που μπορεί να παράγει η εντολή ping είναι, επομένως, 65.528 bytes.

4.11

Όχι δεν έχει μεταφερθεί ως ένα πακέτο IPv4, αφού είναι πολύ μεγάλο και απαιτεί θρυμματισμό.

4.12

ip.addr == 192.168.1.7						
No.	Time	Source	Destination	Protoc	Length	Info
51	0.000000	192.168.1.4	192.168.1.7	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=85f5) [Reassembled in #55]
52	0.000000	192.168.1.4	192.168.1.7	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=85f5) [Reassembled in #55]
53	0.000000	192.168.1.4	192.168.1.7	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=85f5) [Reassembled in #55]
54	0.000000	192.168.1.4	192.168.1.7	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=85f5) [Reassembled in #55]
• 55	0.000000	192.168.1.4	192.168.1.7	ICMP	122	Echo (ping) request id=0x0001, seq=93/23808, ttl=64 (reply in 61)
57	0.021247	192.168.1.7	192.168.1.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=4a44) [Reassembled in #61]
58	0.000982	192.168.1.7	192.168.1.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=4a44) [Reassembled in #61]
59	0.000946	192.168.1.7	192.168.1.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=4a44) [Reassembled in #61]
60	0.001068	192.168.1.7	192.168.1.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=4a44) [Reassembled in #61]
61	0.000000	192.168.1.7	192.168.1.4	ICMP	122	Echo (ping) reply id=0x0001, seq=93/23808, ttl=64 (request in 55)

Όπως φαίνεται και στο Screenshot, αφού το μέγεθος ήταν 6000 bytes, τότε το πακέτο θα σπάσει σε 5 πακέτα (αφού το μέγιστο πακέτο που στέλνεται χωρίς θρυμματισμό είναι 1472 bytes. Κάθε πλαίσιο από τα πρώτα 4 έχει συνολικό μέγεθος 1514 bytes, όπου 14 bytes είναι το Ethernet header, 20 bytes είναι το IP header και 1480 είναι το ICMP πακέτο. Το τελευταίο πλαίσιο έχει μέγεθος 122 bytes, από τα οποία πάλι 14 αντιστοιχούν στο Ethernet header και 20 στο IP header. Επιπλέον, 8 bytes αντιστοιχούν στο ICMP header, άρα μένουν 80 bytes. Συνεπώς, $4 \cdot 1480 + 80 = 6000$ bytes.

4.13

- **Θραύσμα 51:** Identification = 0x85f5, Don't Fragment Bit = 0 (Not set), More Fragments Bit = 1 (Set), Fragment Offset = 0
- **Θραύσμα 52:** Identification = 0x85f5, Don't Fragment Bit = 0 (Not set), More Fragments Bit = 1 (Set), Fragment Offset = 1.480
- **Θραύσμα 53:** Identification = 0x85f5, Don't Fragment Bit = 0 (Not set), More Fragments Bit = 1 (Set), Fragment Offset = 2.960
- **Θραύσμα 54:** Identification = 0x85f5, Don't Fragment Bit = 0 (Not set), More Fragments Bit = 1 (Set), Fragment Offset = 4.440
- **Θραύσμα 55:** Identification = 0x85f5, Don't Fragment Bit = 0 (Not set), More Fragments Bit = 0 (Not Set), Fragment Offset = 5.920

4.14

Από το 2ο bit του πεδίου Flag (Don't Fragment) της επικεφαλίδας IPv4 μπορούμε να καταλάβουμε ότι αν είναι 0 σημαίνει πως το πακέτο μπορεί να έχει θρυμματιστεί. Μπορούμε όμως από το 3ο bit του πεδίου Flag (More Fragments) να καταλάβουμε σίγουρα ότι αποτελεί θραύσμα, αν έχει την τιμή 1.

4.15

Η τιμή του πεδίου Fragment Offset της επικεφαλίδας IPv4, αν έχει τιμή 0 δηλώνει ότι το συγκεκριμένο θραύσμα είναι πρώτο κατά σειρά.

4.16

Το συνολικό μήκος του θραύσματος (frame length) είναι 1514 bytes.

4.17

Όμοια, Η τιμή του πεδίου Fragment Offset της επικεφαλίδας IPv4, αν έχει τιμή μεγαλύτερη του 0 δηλώνει ότι το συγκεκριμένο θραύσμα είναι μεταγενέστερο και στην περίπτωσή μας συγκεκριμένα αποτελεί το δεύτερο κατά σειρά θραύσμα.

4.18

Ναι, ακολουθούν και άλλα θραύσματα.

4.19

Από το 3ο bit του πεδίου Flag (More Fragments) μπορούμε να καταλάβουμε ότι σίγουρα ακολουθούν και άλλα θραύσματα, αν αυτό έχει τιμή 1 (Set).

4.20

Τα πεδία της IPv4 επικεφαλίδας, τα οποία διαφοροποιούνται μεταξύ πρώτου και δεύτερου θραύσματος είναι τα πεδία Fragment Offset και Header Checksum.

4.21

Το προτελευταίο θραύσμα, έχει στο πεδίο Fragment Offset τιμή 4.440, πράγμα που είναι αναμενόμενο αφού ο αριθμός αυτός ισούται με όσα δεδομένα (σε bytes) ICMP έχουν αποσταλεί πριν από αυτό το θραύσμα ($3 \cdot 1.480 = 4.440$). Έτσι και το τελευταίο θραύσμα, στο πεδίο Fragment Offset τιμή 5.920, αφού έχουν αποσταλεί προηγουμένως 4 θραύσματα μεγέθους 1.480 bytes (συνολικό ICMP μέγεθος), άρα $4 \cdot 1.480 = 5.920$.

4.22

Κάθε θραύσμα έχει διαφορετικό Header Checksum καθώς και διαφορετικό Fragment Offset. Ακόμη, όσον αφορά το πεδίο Flag, τα 2 πρώτα θραύσματα έχουν τιμή 0x20, το 3ο 0x21, το 4ο 0x22, ενώ το τελευταίο 0x02. Τέλος, τα 4 πρώτα θραύσματα έχουν το 3ο bit του πεδίου Flag (More Fragments) ίσο με 1 (Set), ενώ το τελευταίο ίσο 0 (Not set).