Ονοματεπώνυμο: Ανδρέας Στάμος (03120***)	**Ομάδα:** 1
Όνομα PC/ΛΣ: linux / Ubuntu 22.04.2 LTS (με VPN στο δίκτυο του Πολυτεχνείου)	**Ημερομηνία:** 07/11/2023
Διεύθυνση IP: 147.102.131.218	**Διεύθυνση MAC:** DE–3F–DC–B2–E0–D0

Εργαστηριακή Άσκηση 6
Πρωτόκολλο ICMP

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

Η εργασία υλοποιήθηκε με σύνδεση στο δίκτυο VPN του Πολυτεχνείου.

Άσκηση 1

1.1 `ether host de:3f:dc:b2:e0:d0`

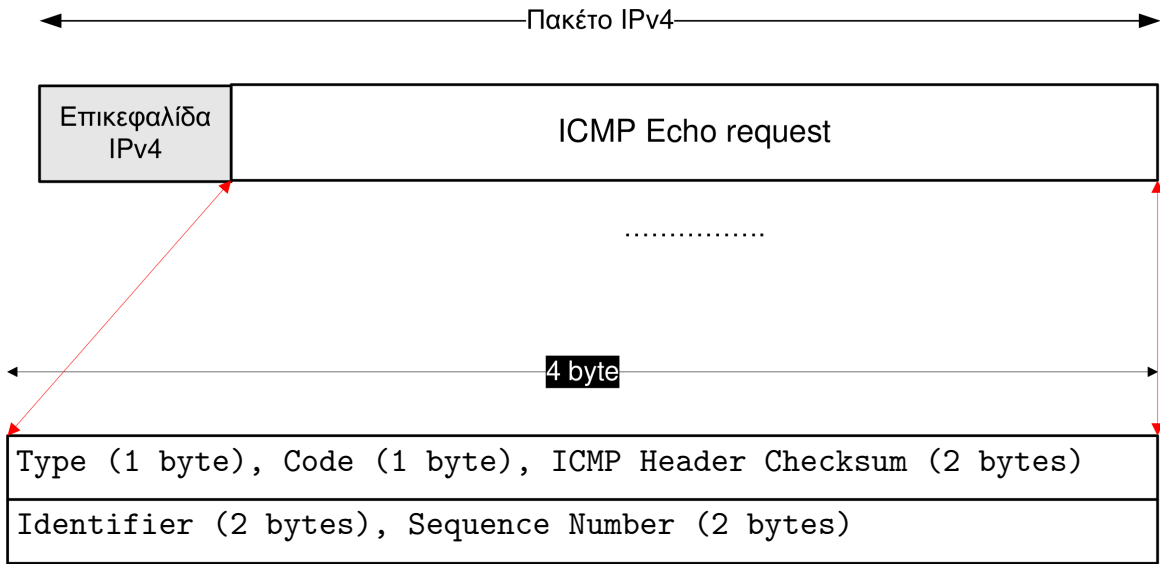
1.2 `arp or icmp`

1.3 Δεν καταγράφηκαν

1.4 Protocol με τιμή 0x01 για το ICMP

1.5 8 bytes

1.6



1.7 Type: 0x08, Code: 0x00

1.8 Identifier: 0x000a, Sequence number: 0x0001

1.9 Τα παρακάτω αναφέρονται στο `ping` του πακέτου `iputils` σε Linux και δεν καθορίζονται από το πρότυπο ICMP.

To default ICMP Data Size είναι 56 bytes.

Αρχικά αποθηκεύεται στο payload η τρέχουσα χρονική στιγμή σε Unix Time (δευτερόλεπτα και μικροδευτερόλεπτα από το Epoch, που είναι 1970-01-01 00:00:00 +0000 UTC). Πιο συγκεκριμένα το `ping` καλεί το Linux syscall `gettimeofday` που επιστρέφει ένα struct τύπου `timeval` και έπειτα αντιγράφει την αναπαράσταση στην μνήμη του struct αυτού στο payload. Η αντιγραφή γίνεται byte-byte, γεγονός που μπορεί να επαληθεύσει κανείς παρατηρώντας στο Wireshark πως οι δύο αριθμοί (δευτερόλεπτα και μικροδευτερόλεπτα

– 8-bytes ακέραιοι καθένα στην έκδοση του Linux και CPU που διαθέτω), έχουν γραφτεί σε little-endian σειρά όπως ισχύει για την x86 CPU του υπολογιστή μου.

1.10 8 bytes, ίδια δομή με του Echo Request.

1.11 Type: 0x00, Code: 0x00

1.12 Type

1.13 Type: 0x08, Code: 0x00

1.14 Ήδη στο 1.13 επέλεξα το Echo Reply στο Echo Request του 1.8.

Το Echo Reply έχει το ίδιο Identifier και Sequence Number με το Echo Request στο οποίο απαντάει.

1.15 Σύμφωνα με το πρότυπο ICMP, τα Identifier και Sequence Number ταυτοποιούν μοναδικά ένα Echo Request ώστε τα Echo Replies να μπορούν να αντιστοιχηθούν 1-1 με τα Echo Requests.

Συγκεκριμένα στο `ping` του πακέτου `iputils` σε Linux, το Identifier ταυτοποιεί μοναδικά την διεργασία `ping` που έστειλε το Echo Request και το Sequence Number είναι αύξων αριθμός των Echo Requests που έχουν σταλεί ως τώρα από την διεργασία. Το Identifier ταυτοποιεί την διεργασία, προκειμένου διαφορετικές διεργασίες `ping` να μπορούν να στείλουν Echo Requests προς τον ίδιο target host και όταν έρθουν τα Replies να μπορούν να διακρίνουν ποια Replies αφορούν ποια διεργασία – και οι δύο διεργασίες θα ξεκινήσουν από Sequence Number 1.

Ακόμα πιο συγκεκριμένα, το `ping` λετουργεί από προεπιλογή με Linux sockets τύπου `SOCK_DGRAM` με υλοποίηση του ICMP από το Linux. Αν για κάποιο λόγο αυτό αποτύχει, κάνει fallback σε sockets τύπου `SOCK_RAW`, οπότε αναγκάζεται να ορίσει όλες τις λεπτομέρειες του ICMP μόνο του. Στην περίπτωση αυτή θέτει στο Identifier to `rand() & 0xFFFF` δηλαδή έναν τυχαίο 2-byte ακέραιο.

Στην περίπτωση όμως που το Linux αναλάβει την υλοποίηση του ICMP (στην default περίπτωση δηλαδή) μελετώντας τα αρχεία κώδικα του Linux `net/ipv4/icmp.c` και `net/ipv4/ping.c` βρίσκουμε ότι το Linux αναζητά ένα Identifier που να μην χρησομοποιείται ήδη από άλλο socket που επίσης να εκτελεί `ping`. Αυτό το κάνει αποθηκεύοντας τον τελευταίο Identifier που εκχώρησε (μεταβλητή `ping_port_rover`) και αυξάνοντας από την τιμή αυτή ένα-ένα και ελέγχοντας σε έναν πίνακα (`ping_table`) όπου είναι αποθηκεύμενα όλα τα ανοιχτά sockets για pings που έχουν γίνει.

1.16 56 bytes

1.17 Όχι, το περιεχόμενο του Echo Reply για ένα συγκεκριμένο Echo Request (ίδιο Identifier και Sequence Number) είναι ίδιο με περιεχόμενο του αντίστοιχου Echo Request. Μάλιστα, η συμπεριφορά αυτή είναι καθορισμένη και στο πρότυπο του ICMP, το RFC 792.

Συμπλήρωση: Στην πραγματικότητα, το `ping` του πακέτου `iputils` βασίζεται σε αυτήν την ιδιότητα για να υπολογίζει το RTT, καθώς στέλνει στο Echo Request την χρονική στιγμή που έστειλε το πακέτο (βλ. ερώτημα 1.9) και αναμένει υποχρεωτικά στο Echo Reply να λάβει το δεδομένο αυτό, δηλαδή την ώρα που απεστάλη το Echo Request. Μόλις λάβει το Echo Reply, αφαιρεί την τρέχουσα χρονική στιγμή από την στιγμή που έλαβε στο περιεχόμενο του Echo Reply και έτσι βρίσκει το RTT. Κατ' αυτόν τον τρόπο, το `ping` αποφεύγει να αποθηκεύει σε πίνακες τις χρονικές στιγμές που έστειλε ένα Echo Request με ορισμένο Identifier και Sequence Number και επίσης αποφεύγει όταν έρθει ένα Echo Reply να έπρεπε να κάνει αναζήτηση σε αυτόν τον πίνακα όταν έρθει το Echo Reply, για το πότε απεστάλη το αντίστοιχο Echo Request. Αυτό μπορεί σήμερα να φαίνεται σαν χώρος και χρόνος υπολογισμού αμελητέο, αλλά όταν γράφτηκε το `ping` μάλλον ήταν σημαντικοί υπολογιστικοί όροι για τους υπολογιστές της τότε εποχής.

1.18 Το `ping` εκτυπώνει όταν λάβει Echo Reply για ένα Echo Request που απέστειλε. Εκτυπώνει το συνολικό μέγεθος του πακέτου ICMP (ICMP data + ICMP header χωρίς IP header), το Sequence Number και το RTT που υπολογίζει με τον τρόπο που σχολιάζεται στα ερωτήματα 1.15 και 1.17.

1.19 `ping -c 2 {TARGET IP ADDRESS}`

1.20 3

1.21 Μετράμε περίπου κάθε 1 sec. Το Linux τα στέλνει ανά `retrans_time_ms` που έχει default τιμή 1000 ms

Βλ. και ερώτημα 2.6 σχετικά με την υλοποίηση του ARP στο Linux

1.22 Τα Echo Requests δεν αποστέλλονται ποτέ. Ο υπολογιστής γνωρίζει πως η διεύθυνση IP βρίσκεται στο ίδιο υποδίκτυο, οπότε προσπαθεί μέσω ARP Request να βρεί την MAC Address της IP διεύθυνσης στην οποία θέλει να στείλει. Το ARP Request δεν το απαντάει κανένας, οπότε η ARP εγγραφή μπαίνει σε κατάσταση `FAILED`, οπότε ο host θεωρείται πως είναι down.

Με την καταγραφή του Wireshark στην κανονική κάρτα δικτύου, παρατηρούμε επομένως μόνο τα ARP Requests. Ωστόσο, αν με το Wireshark καταγράψουμε και την κίνηση στην loopback συσκευή, θα παρατηρήσουμε πως το Linux στέλνει στο loopback, ένα ICMP Destination Host Unreachable.

1.23 Το `ping` λαμβάνει από τον loopback οδηγό τα ICMP Destination Host Unreachable και τα εκτυπώνει. (έχουν Identifier που αντιστοιχεί στο socket από όπου παράδοθηκαν τα Echo Requests στο Linux για αποστολή στο δίκτυο – οπότε τα λαμβάνει – και Sequence Number το Sequence Number του Echo Request που δεν κατάφερε να αποσταλή).

Άσκηση 2

2.1 `147.102.200.200`, που είναι η διεύθυνση της προκαθορισμένης πύλης

2.2 Αποστολέας: de:3f:dc:b2:e0:d0
Παραλήπτης: 08:ec:f5:d0:d9:1d

2.3 Αποστολέας: 147.102.200.37
Παραλήπτης: 147.102.1.1

2.4 Η διεύθυνση MAC αποστολέα αντιστοιχεί στην διεύθυνση IP του υπολογιστή μου.
Η διεύθυνση MAC παραλήπτη αντιστοιχεί στην διεύθυνση IP της προκαθορισμένης πύλης (`147.102.200.200`).

2.5 Ναι.

2.6 Στο Linux οι εγγραφές στην ARP cache παραμένουν σε κατάσταση `REACHABLE` (δηλαδή έγκυρες) για χρόνο ίσο με μια τυχαία τιμή ανάμεσα στις `base_reachable_time_ms/2` και `3*base_reachable_time_ms/2`, και έπειτα πάνε σε κατάσταση `STALE`.

Αν μια ARP εγγραφή είναι σε κατάσταση `STALE` και γίνει πρόσβαση σε αυτή (για την μετάφραση της IP διεύθυνσής της), η ARP cache παραδίδει την διεύθυνση MAC που είχε ήδη, και βάζει την εγγραφή σε κατάσταση `DELAY`, όπου παραμένει για χρόνο `delay_first_probe_time`. Στην συνέχεια η εγγραφή μπαίνει σε κατάσταση `PROBE`. Αποστέλλονται ανά χρονικά διάστηματα `retrans_time_ms` πακέτα ARP Requests. Αποστέλλονται το πολύ `mcast_solicit` πακετά ARP Requests και αν δεν έρθει απάντηση σε κανένα από αυτά η εγγραφή μπαίνει σε κατάσταση `FAILED`. Αν έρθει απάντηση ARP Reply, τότε η εγγραφή μπαίνει ξανά σε κατάσταση `REACHABLE`. Σημειώνεται πως μόνο στην κατάσταση `FAILED` η ARP δεν παραδίδει διεύθυνση MAC στα ανώτερα στρώματα δικτύου, καθώς ο host θεωρείται ότι είναι πλεόν down. Στις καταστάσεις `STALE`, `DELAY`, `PROBE` παραδίδεται στα ανώτερα στρώματα δικτύου κανονικά η υπάρχουσα MAC διεύθυνση.

Ο λόγος που υπάρχει η κατάσταση `DELAY` είναι πως ανώτερα στρώματα δικτύου μπορούν να επιβεβαιώσουν πως η παλιά αντιστοιχία IP-MAC διευθύνσεων εξακολουθεί να ισχύει, οπότε η ARP εγγραφή πηγαίνει ξανά σε κατάσταση `REACHABLE` χωρίς ARP Request. Για παράδειγμα, ενδέχεται να πρόκειται για TCP σύνδεση (ιδιαίτερα συχνό) και να ληφθεί απάντηση ACK. Αν συμβεί αυτό γνωρίζουμε ότι η παλιά αντιστοιχία είναι σωστή οπότε δεν απαιτείται να γίνει νέο ARP Request. Για το TCP υλοποιείται από το Linux απευθείας. Ο τρόπος με τον οποίο στο Linux, μπορούμε να ενημερώσουμε την ARP cache πως η ARP εγγραφή είναι εγκυρή, είναι όταν στέλνουμε ένα πακέτο και αν το socket είναι τύπου `SOCK_RAW` ή `SOCK_DGRAM` να θέσουμε στο syscall `send()` το flag `MSG_CONFIRM`.

Παραδόξως όταν λαμβάνεται ICMP Echo Reply και ενώ η ARP εγγραφή βρίσκεται στην κατάσταση `DELAY`, ενώ το λογικό θα ήταν να ανανεώνεται απευθείας σε `REACHABLE` μόλις ληφθεί το Echo Reply, αυτό δεν συμβαίνει, η εγγραφή μπαίνει σε κατάσταση `PROBE` και αποστέλλεται ARP Request. Ενδεχομένως, επειδή το `ping` χρησιμοποιείται συνήθως για διαγνωστικούς σκοπούς, αυτή η συμπεριφορά να έχει σχεδιαστεί επίτηδες.

Έτσι **αυτό που παρατηρούμε τελικά είναι** πως ανά χρόνο με μέση τιμή `base_reachable_time_ms` – η τιμή επιλέγεται τυχαία όπως αναφέραμε παραπάνω – συμβαίνει το εξής:
Η ARP εγγραφή μπαίνει σε κατάσταση `STALE`. Το `ping` επιχειρεί να στείλει νέο Echo Request. Η ARP cache του παραδίδει την παλιά MAC διεύθυνση και βάζει την ARP εγγραφή σε κατάσταση `DELAY`. Συμβαίνει μια αναμονή χρόνου `delay_first_probe_time`. Έπειτα η εγγραφή μπαίνει σε κατάσταση `PROBE`, αποστέλλεται ένα ARP Request, λαμβάνεται ένα ARP Reply και η ARP εγγραφή ξαναμπαίνει σε κατάσταση `REACHABLE`.

Οι default τιμές στο σύστημά μου για τις παραπάνω σταθέρες μπορούν να διαβαστούν από τα αρχεία `/proc/sys/net/ipv4/neigh/default/{CONSTANT}`. Σύμφωνα με το manpage `ARP(7)` οι default τιμές είναι:

- `base_reachable_time_ms`: 30000 ms
- `delay_first_probe_time`: 5 sec
- `mcast_solicit`: 3

- `retrans_time_ms`: 1000 ms

2.7 `icmp.type == 0`

2.8 Το TTL που εμφανίζει το `ping` είναι το TTL του IP πακέτου του Echo Reply. (ισούται με την διαφορά του πλήθους δρομολογητών που μεσολάβησαν από το προορισμό μέχρι εμάς από την αρχική τιμή TTL που έθεσε ο προορισμός)

2.9 Μόνο τα Echo Requests που αποστέλλει ο υπολογιστής μας

2.10 Όταν πρόκειται για υπολογιστή του τοπικού δικτύου, το Linux γνωρίζει πως πρέπει να τον βρεί απευθείας στο τοπικό δίκτυο απευθείας μέσω του στρώματος ζεύξης δεδομένων. Στέλνει ARP Request και όταν δεν το απαντήσει κανένας, γνωρίζει με βεβαιότητα ότι ο host αυτός είναι down, οπότε παραδίδει μέσω της loopback συσκευής ένα Destination Unreachable.

Όταν όμως πρόκειται για υπολογιστή εκτός του τοπικού δικτύου, το IP πακέτο δρομολογείται μέσω δρομολογητών. Αφενός η IP διεύθυνση μπορεί να μην αντιστοιχεί σε κάποιο υποδίκτυο και κάποιος ενδιάμεσος δρομολογητής να μην ξέρει πως να δρομολογήσει το πάκετο κάνοντας το επομένως drop (απίθανο αφού πρακτικά όλες οι IPv4 έχουν εκχωρηθεί και λογικά όλα τα υποδίκτυα είναι συνδεδεμένα στο διαδίκτυο). Αφετέρου αν η IP διεύθυνση μπορούσε να αντιστοιχεί σε κάποιον host κάποιου υποδικτύου, που υπάρχει και έχει δρομολογητή με το διαδίκτυο, τότε αρμόδιος να απαντήσει με βεβαιότητα ότι ο host είναι down είναι ο τελευταίος δρομολογητής πριν το πακέτο παραδιδόταν στον host, αν αυτός υπήρχε. Ο δρομολογητής αυτός, σύμφωνα με το RFC 792, θα μπορούσε πράγματι να στείλει Destination Unreachable. Όμως δεν έχει λόγο να επιβαρυνθεί άσκοπα καταναλώνοντας πόρους, ενώ επίσης, η απόστολη του Destination Unreachable θα μπορούσε να αποτελεί και απειλή ασφαλείας καθώς αποκαλύπτεται η εσωτερική τοπολογία του δικτύου.

Συνοπτικότερα, όταν πρόκειται για υπολογιστή του τοπικού δικτύου το Destination Unreachable το παραδίδει απευθείας το Linux, ενώ όταν πρόκειται για υπολογιστή εκτός του τοπικού δικτύου, ακόμα και αν γίνεται να απαντηθεί το ερώτημα αν ο host είναι down, αυτό θα έπρεπε να γίνει από κάποιον άλλο, που δεν το κάνει για λόγους ασφαλείας και επιδόσεων.

Μάλιστα, μπορούμε να επιχειρήσουμε και στο τοπικό δίκτυο να στείλουμε IP πακέτο προς διεύθυνση host που είναι down, εξαναγκάζοντας το πακέτο όμως να πάει υποχρεωτικά προς την προκαθορισμένη πύλη. (αυτό γίνεται για παράδειγμα με την εντολή `nping -icmp -e {INTERFACE NAME} --dest-mac {DEFAULT GATEWAY MAC ADDRESS} {TARGET IP ADDRESS}`.) Τότε παρατηρούμε, πως ακόμα και τότε, ότι ο δρομολογητής της προκαθορισμένης πύλης, μας απαντάει με Host Redirect και όχι με Destination Unreachable (αν και βέβαια εδώ πιθανώς αυτό το κάνει για να ξέρουμε ότι αν κάποτε ο host γινόταν up, ή σε κάθε περίπτωση αν θέλαμε να του ξαναστείλουμε, έπρεπε να τον ψάξουμε απευθείας στο τοπικό υποδίκτυο και όχι να στείλουμε στην προκαθορισμένη πύλη).

Άσκηση 3

3.1 Το ICMP Data έχει μέγεθος 32 bytes και περιέχομενο τους αριθμούς σε αύξουσα σειρά από το 0x48 ως το 0x67.

3.2 Στέλνει διαφορετικά δεδομένα από το ping και διαφορετικού μεγέθους. Το `ping` στέλνει τα δεδομένα που στέλνει για τους λόγους που εξηγούνται στο ερώτημα 1.9. Το `traceroute` δεν φαίνεται να έχει συγκεκριμένο λόγο που στέλνει το range από 0x48 ως 0x67. Πάντως σύμφωνα με μια πρόχειρη μελέτη στο διαδίκτυο, φαίνεται να στέλνει συγκεκριμένα αυτό και όχι τυχαία δεδομένα, ώστε η κίνηση που προέρχεται από `traceroute` να διακρίνεται εύκολα για διαγνωστικούς σκοπούς.

3.3 ICMP TTL exceeded

3.4 Type=10=0χ0b και Code=0

3.5 Checksum (2 bytes), και Length (1 byte)

3.6 Συνολικά η επικεφαλίδα ICMP είναι 8 bytes. Υπάρχουν 3 bytes που δεν χρησιμοποιούνται (Unused) – είναι το 5ο, το 7ο και το 8ο byte.

3.7 Το ICMP Data περιέχει το IP πακέτο το οποίο έγινε drop επειδή απέκτησε TTL=0 (zero-padded ώστε να έχει μέγεθος πολλαπλάσιο των 32-bits αφού το Length εκφράζεται σε πλήθος 32-bits λέξεων.)

Άσκηση 4

4.1 Μέγεθος IP πάκετου = Επικεφαλίδα IP + Επικεφαλίδα ICMP + ICMP Data = 20 bytes + 8 bytes + ICMP Data

Άρα: Μέγεθος ICMP Data = Μέγεθος IP πακέτου − 28 bytes = MTU − 28 bytes

Με άλλα λόγια π.χ. για MTU 1492 bytes πρέπει να στείλουμε ICMP Data μεγέθους 1492−28 = 1464 bytes.

Για την αποστολή των πακέτων συντάχθηκε το Bash script:

```
for MTU in 65535 32000 17914 8166 2002 1492 1006 508 296 68
do
        ping -W 1 -c 1 -M do -4 -s $(($MTU+28)) 147.102.40.15
done

```

4.2 **Προσοχή:** Το Linux, σύμφωνα και με το πρότυπο RFC 1191, αποθηκεύει τα MTU που ανακαλύπτει. Το RFC 1191 απαγορεύει ρητά να γίνει ξανά ελέγχος PMTU πριν περάσουν 5 λεπτά από τον προηγούμενο, οπότε το Linux δεν έχει λόγο να μας επιτρέπει "εύκολα" να το κάνουμε νωρίτερα. Το Linux αποθηκεύει τις πληροφορίες για το MTU για έναν συγκεκριμένο host στην Forwarding information Base – που στο Linux έχει δομή δεδομένων ως Trie (δέντρο προθεμάτων) στην μνήμη – βλ. και https://www.kernel.org/doc/Documentation/networking/fib_trie.txt και https://en.wikipedia.org/wiki/Luleå_algorithm.

Στα πλαίσια αυτά, για να μπορέσουμε να παρατηρήσουμε την διαδικασία PMTU πρέπει να εκκαθαρίσουμε την cache. Η διαγραφή ενός μόνο entry της FIB δεν φαίνεται να μπορεί να γίνει στο Linux. Μπορούμε όμως να διαγράψουμε την πλήρη FIB με την εντολή: `ip route flush cache`.

Σημειώνεται ότι η εκτέλεση PMTU πριν περάσουν 5 λεπτά από την προηγούμενη φορά που έγινε απαγορεύεται ρητά από το RFC 1191 και δυνητικά μπορεί να προκαλέσει άγνωστες συνέπειες – π.χ. κάποιος δρομολογητής να το ερμηνεύσει ως DoS επίθεση, αν και αυτό, για φυσιολογικούς χρόνους, δεν θεωρείται πιθανό καθώς δυνητικά θα μπορούσαμε να είχαμε κάνει restart τον υπολογιστή πριν περάσουν 5 λεπτά.

Συμπλήρωση: Μπορούμε να δούμε την τρέχουσα εγγραφή στην FIB για μια συγκεκριμένη διεύθυνση με την εντολη: `ip route get {TARGET IP ADDRESS}`. (φαίνεται το τρέχον MTU – αν υπάρχει – και ο χρόνος που η τιμή αυτή θα λήξει)

Χωρίς να γνωρίζει το Linux το MTU για την 147.102.40.15 τρέχουμε τα `ping` για την PMTU. (βλ. Bash script στο ερώτημα 4.1) Παραδόξως, υπάρχουν τιμές του MTU για τις οποίες δεν λαμβάνουμε Echo Reply αλλά ούτε και Destination Unreachable (Fragmentation Needed). Η συμπεριφορά αυτή είναι παράξενη, διότι κάποιος θα μπορούσε να παρερμηνεύσει τον host ως down αντί να εκτελέσει PMTU Discovery. Δεν φαίνεται να προβλέπεται από το RFC 1191 τέτοια συμπεριφορά. Δοκιμάζοντας διευθύνσεις στο ίδιο υποδίκτυο (π.χ 147.102.40.14 και 147.102.40.16 – θέτοντας την σημαία RECORD ROUTE επιβεβαιώνουμε ότι είναι στο ίδιο υποδίκτυο) οι υπόλοιποι hosts του υποδικτύου φαίνεται να υποστηρίζουν MTU μεγαλύτερο από 576 bytes. Συνεπώς, λογικά, το μειωμένο MTU αφορά αποκλειστικά τον host 147.102.40.15. Πάντως, το υποδίκτυο του Πολυτεχνείου είναι τύπου Ethernet, οπότε το μειώμενο MTU αποκλειστικά σε έναν host και μάλιστα σε τιμή 576 bytes δεν φαίνεται να δικαιολογείται. Ίσως με software ο host απορρίπτει πακέτα μεγαλύτερα από 576 bytes (ή ίσως αυτό συμβαίνει από κάποιο switch).

Σε κάθε περίπτωση, αν θέλουμε να δούμε το Destination Unreachable (Fragmentation Needed) μπορούμε να βάλουμε MTU μεταξύ 1500-8=1492 και 1500 και να στείλουμε σε κάποιον κόμβο εκτός του τοπικού υποδικτύου! Ο λόγος που αυτό συμβαίνει είναι επειδή ο τροπός που, σε μια τυπική οικιακή εγκατάσταση, το υποδίκτυο συνδέεται με το δίκτυο του ISP είναι μέσω PPPoE οπότε υπάρχουν 8 bytes επικεφαλίδα του PPPoE. (Δηλαδή ο δρομολογητής της οικιακής μας εγκατάστασης είναι αυτός που θα μας στείλει το Destination Unrechable).

Συμπλήρωση: Ο τρόπος που εμφανίζεται το PPPoE είναι ο εξής:

Ο υπολογιστής μας στέλνει IP πακέτα προς την προκαθορισμένη πύλη. Η προκαθορισμένη πύλη ενθυλυκώνει τα IP πακέτα σε πλαίσια PPPoE και τα προωθεί προς τον ISP – μέσω του DSL. Ο ISP αναλαμβάνει να αποενθυλακώσει τα PPPoE πλαίσια και να τα προωθήσει στο IP δίκτυό του κατάλληλα. Ένας από τους λόγους που χρησιμοποιείται το PPPoE είναι προκειμένου ο ISP να μπορεί να παρέχει διαφορετικές υπηρεσίες σε διαφορετικούς συνδρομητές.

TL;DR Είτε με τον παραπάνω τρόπο, είτε με την δοθείσα καταγραφή, παρατηρούμε Destination Unreachable (Fragmentation Needed). Στην περίπτωση που περιγράφουμε παραπάνω, το πακέτο μάς το στέλνει η προκαθορισμένη πύλη (της οικιακής μας εγκατάστασης).

4.3 Type=0x03, Code=0x04

4.4 Το Code δηλώνει πως το λάθος οφείλεται στην απαίτηση μη θρυμματισμού του IPv4.

Το Next-Hop MTU έχει τιμή 1492 bytes. (ταυτίζεται με την τιμή που σχολιάστηκε στο ερώτημα 4.2)

4.5 Το πακέτο περιέχει τα πρώτα 548 bytes του IP πακέτου που έγινε drop. **Σημείωση:** Στο πακέτο Destination Unreachable που λάβαμε δεν υπάρχει πεδίο Length (οι αντίστοιχες θέσεις είναι μηδενικές). Σε πρώτη σκέψη, σύμφωνα με το πρότυπο RFC 4884, για το οποίο γίνεται λόγος στην εκφώνηση, για να λάβουμε 548 bytes δεδομένων έπρεπε να υπάρχει αντίστοιχη τιμή στο Length. Σκεφτόμαστε ότι αφού δεν υπάρχει, η διαδικασία θα έπρεπε να γίνεται σύμφωνα με το αρχικό πρότυπο RFC 792, σύμφωνα με το οποίο το Destination Unreachable περιέχει την IP επικεφαλίδα μαζί με τα 8 πρώτα bytes των δεδομένων του πακέτου που έγινε drop. Ούτε αυτό όμως συμβαίνει. Μεταξύ, όμως, του έτους 1981 που ορίστηκε το RFC 792 και του έτους 2007 που ορίστηκε το RFC 4884, το έτος 1995 είχε οριστεί το RFC 1812, το οποίο στην παράγραφο 4.3.2.3 όριζε πως τα ICMP πακέτα πρέπει να περιέχουν όσο δυνατόν περισσότερο από το πακέτο που έγινε drop χωρίς το μέγεθος του ICMP πακέτου να ξεπεράσει τα 576 bytes (υπενθυμίζεται ότι σύμφωνα με το IP πρωτόκολλο 576 bytes είναι το ελάχιστο μέγεθος πακέτου το οποίο όλοι οφείλουν να αποδέχονται). Μάλιστα, αυτή ακριβώς είναι η περίπτωση που συμβαίνει εδώ, αφού το ICMP πακέτο που λαμβάνουμε έχει μέγεθος 576 bytes.

4.6 Ακολουθώντας την προβλεπόμενη διαδικασία για PMTU του RFC 1191, πετυχαίνουμε ακριβώς την τιμή 1492. (εξάλλου και οι συγγραφείς στον σχετικό πίνακα αναφέρουν πως την έχουν διάλεξει λόγω του MTU του PPP)

4.7 Για MTU μεταξύ των τιμών 1492 και 577 bytes δεν λαμβάνουμε Destination Unreachable ούτε Echo Reply. Ξανασημειώνουμε ξανά ότι αυτό είναι παράξενο (το λογικό είναι ο αποστολέας να θεωρήσει τον host down και όχι να κάνει PMTU Discovery).

4.8 576

4.9 Ο ISP προωθεί κανονικά τα πακέτα μεγέθους $\leq$ 1492 bytes. Επίσης το υποδίκτυο του Πολυτεχνείου προωθεί τα πακέτα μεγέθους $\leq$ 1492 bytes. Hosts στο ίδιο υποδίκτυο (βλ. ερώτημα 4.2) λαμβάνουν και απαντάνε σε πακέτα μεγέθους $\leq$ 1492 bytes. Συνεπώς το μειωμένο MTU των 576 έχει να κάνει με την διεπαφή του τελικού host (ή ακριβέστερα με την σύνδεση του τελευταίου δρομολογητή με τον τελικό host – μπορεί να φταίει κάποιο ενδιάμεσο switch)

4.10 Ναι, και μάλιστα αν στείλουμε Echo Request χωρίς την σημαία Don't Fragment απαντά επίσης χωρίς την σημαία αυτή. Η συμπεριφορά αυτή δεν φαίνεται να είναι ορισμένη από τα πρότυπα, αλλά μάλλον καθορίζεται από την υλοποίηση του λειτουργικού συστήματος του host 147.102.40.15.

(Είναι λογική συμπεριφορά. Γενικά το Don't fragment προτιμάται. Όμως αν για κάποιο λόγο ο αποστολέας ζητήσει να επιτρέπεται fragmentation, μάλλον και το πακέτο απάντησης, αφού μάλλον θα ακολουθήσει την ίδια διαδρομή, φαίνεται να είναι καλή ιδέα να του επιτρέπεται fragmentation. Αν το πακέτο του αποστολέα υπέστη fragmentation, αυτό θα απαιτηθεί μάλλον και για το πακέτο απάντησης, αν ακολουθήσει την ίδια διαδρομή)

4.11 Ο τελικός προορισμός δεν θα μπορούσε να λάβει πακέτα μεγέθους μεγαλύτερου από την MTU της διεπαφής του, αφού εξ ορισμού της MTU, θα ήταν τεχνικά αδύνατο να φτάσουν σε αυτόν. Γενικότερα, η απόρριψη πακέτων χωρίς να φτάνει Destination Unreachable στον αποστολέα είναι παράξενη. Ο τρόπος που έχει συνταχθεί το ερώτημα υποδηλώνει πως τα πακέτα φτάνουν στον host και γίνονται drop αφού φτάσουν. Ο λόγος που αυτό συμβαίνει παραμένει άγνωστος. Ίσως γίνονται drop μεγάλα πλαίσια απευθείας από την κάρτα δικτύου για λόγους hardware ή ίσως γίνονται drop με software τρόπο (πιθανώς για διδακτικούς λόγους).

4.12 Όχι. Είναι ο αμέσως μικρότερος αριθμός του MTU που γράφεται ως (πολλαπλάσιο του 8 bytes + 20 bytes).

Ο λόγος που συμβαίνει αυτό είναι διότι το fragment offset εκφράζεται ως πολλαπλάσιο λέξεων των 32 bytes. Οπότε το τμήμα του IP data πρέπει να έχει μέγεθος πολλαπλάσιο του 8 bytes. Επίσης μεταδίδεται και 20 bytes επικεφαλίδα IP.

Άσκηση 5

5.1 `host 147.102.40.15`

5.2 `dig @147.102.40.15 edu-dy.cn.ntua.gr`

5.3 `;; communications error to 147.102.40.15#53: connection refused`

Το μήνυμα λάθους σημαίνει ότι ο host δεν δέχτηκε το πακέτο που του στείλαμε στην συγκεκριμένη θύρα.

5.4 Ναι, ο υπολογιστής μας απέστειλε προς τον 147.102.40.15 ένα DNS query.

5.5 UDP στη θύρα 53

5.6 Ναι

5.7 Type=0x03, Code=0x03

5.8 Code

5.9 Έχουν μεταδοθεί τα πρώτα 28 bytes του IP πακέτου που έγινε drop. Έτσι μπορούμε να δούμε την UDP επικεφαλίδα του πακέτου που έγινε drop και να δούμε ότι είχε θύρα προορισμού την 53, που είναι του DNS.

5.10 Destination Unreachable (Port Unreachable)

Άσκηση 6

6.1 `ping -6 2001:648:2000:329::101`
`traceroute -I -6 2001:648:2000:329::101`

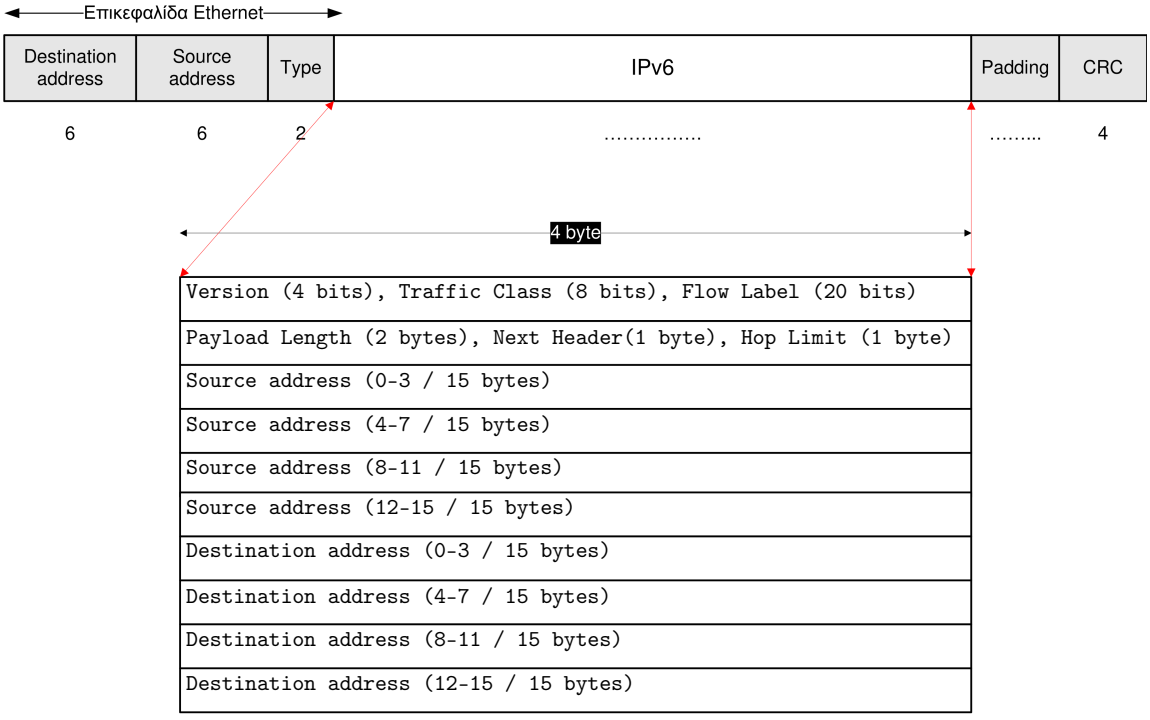
6.2 Capture Filter: `ip6`

Display Filter: `icmpv6`

6.3 0x86dd

6.4 40 bytes

6.5



6.6 Hop Limit

6.7 Next Header με τιμή για το ICMPv6 να είναι 0x3a=58

6.8 Η δομή της επικεφαλίδας είναι ίδια.

6.9 Type=0x80=128 και το πακέτο μεταφέρει 56 bytes δεδομένων.

6.10 Ναι

6.11 Type=0x81=129 και το πακέτο μεταφέρει 56 bytes δεδομένων.

6.12 Περιέχει δεδομένα μεγέθους 32 bytes αντί 56 bytes.

6.13 Όχι. Στο ICMPv6 TTL Exceeded το Length βρίσκεται στο 5ο byte ενώ στο ICMP TTL Exceeded βρίσκεται στο 6ο byte.

6.14 Type=0x03 και το πακέτο μεταφέρει 80 bytes δεδομένων.

6.15 Περιέχει το IPv6 πακέτο που ο ενδιάμεσος δρομολογητής έκανε drop επειδή έφτασε σε τιμή TTL=0.

6.16 Neighbor Solicitation, Router Solicitation (τα ανάλογα του ARP στο IPv6) και Multicast Listener Report Message v2 (για multicast επικοινωνία)

6.17 Neighbor Solicitation: Type=135=0x87, Size=72 bytes
Router Solicitation: Type=133=0x85, Size=56 bytes
Mulicast Listener Report Message v2: Type=143=0x8f, Size=136 bytes