Ονοματεπώνυμο: Ανδρέας Στάμος (03120***)	**Ομάδα:** 1
Όνομα PC/ΛΣ: linux / Ubuntu 22.04.2 LTS (με VPN στο δίκτυο του Πολυτεχνείου)	**Ημερομηνία:** 21/11/2023
Διεύθυνση IP: 147.102.131.218	**Διεύθυνση MAC:** DE–3F–DC–B2–E0–D0

Εργαστηριακή Άσκηση 7
Πρωτόκολλα TCP και UDP

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 `host {HOST IP ADDRESS}`

1.2 `ip.dst in {1.1.1.1, 2.2.2.2, 147.102.40.1}`

1.3 23

1.4 `ip.dst in {1.1.1.1, 2.2.2.2, 147.102.40.1} && tcp.port == 23`

1.5 SYN

1.6 7

Μάλιστα, αυτό αποτελεί την default Kernel Parameter στο Linux (ονομάζεται net.ipv4.tcp_syn_retries). Η default τιμή είναι 6, οπότε γίνονται συνολικά 1 αρχική προσπάθεια + 6 επαναπροσπάθειες = 7 προσπάθειες συνολικά.

Μπορούμε να διαβάσουμε την τιμή της παραμέτρου του Πυρήνα με την εντολή `sysctl net.ipv4.tcp_syn_retries` (ή ισοδύναμα `cat /proc/sys/net/ipv4/tcp_syn_retries`) και να ρυθμίσουμε την τιμή της – απαιτούνται δικαιώματα υπερχρήστη (CAP_SYS_ADMIN) – με `sysctl net.ipv4.tcp_syn_retries={DESIRED VALUE}` (ή ισοδύναμα `echo {DESIRED VALUE}>/proc/sys/net/ipv4/tcp_syn_retries`).

1.7 1,2,4,8,16,32 δευτερόλεπτα

1.8 Διαφέρουν στο Source Port (λογικό αφού θα εγκατεστησούσαν διαφορετικές συνδέσεις) και στα Sequence Number και Time Stamp.

1.9 Μόνο το SYN

1.10 Εγκαταλείπει την προσπάθεια.

1.11 `ip.addr == 147.102.40.1`

1.12 1

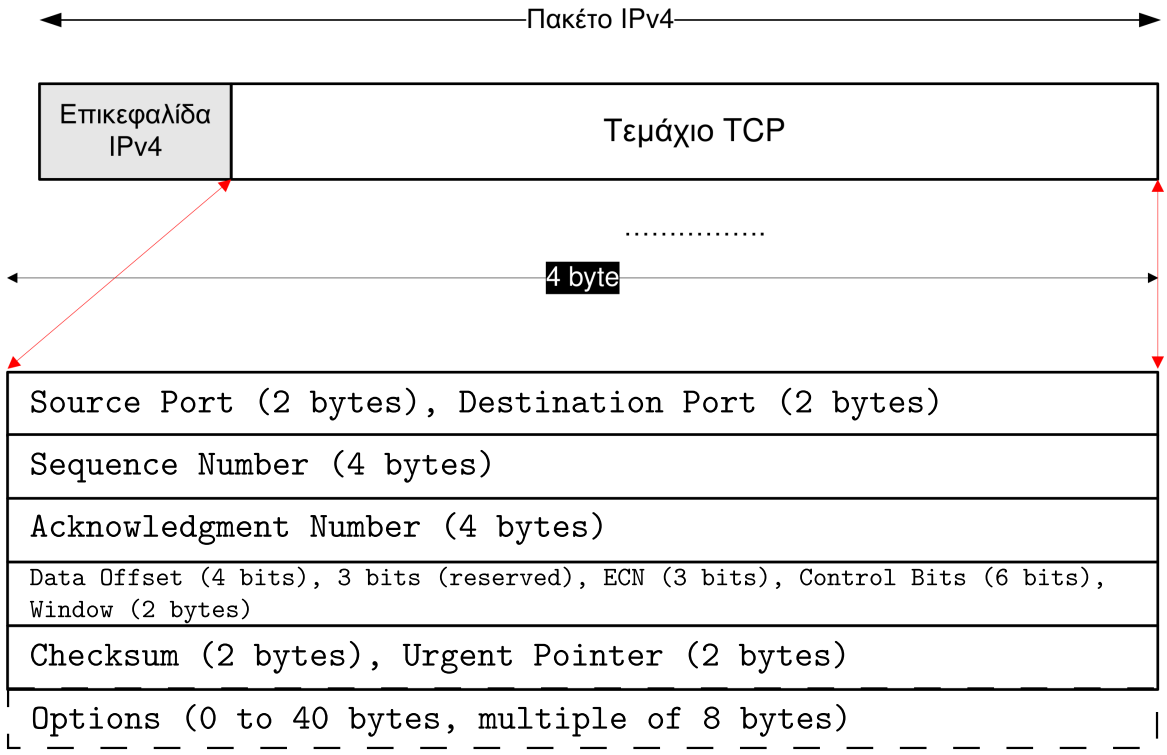
1.13 Ο host 147.102.40.1 μας απαντά στέλνοντας TCP πακέτο με τις σημαίες ACK (δηλαδή πως έλαβε αυτό που του στείλαμε) και RST (δηλαδή πως δεν αποδέχεται την σύνδεση).

1.14 RST και ACK

1.15 RST

1.16 TCP επικεφαλίδα: 20 bytes
TCP δεδομένα: 0 bytes

1.17



1.18 Data Offset και στο Wireshark ονομάζεται Header Length.

1.19 Το Data Offset είναι 4-bit τιμή. Εμείς βλέπουμε το 0x50, όμως μόνο το high nibble, δηλαδή το 0x5 αντιστοιχεί στο Data Offset. Επίσης το Data Offset εκφράζεται ως πλήθος 32-bits (8-byte) λέξεων. Αρα η επικεφαλίδα έχει μέγεθος $0x5 \cdot 4 = 20$ bytes.

1.20 Όχι.

1.21 Μέγεθος Segment = Total Length (IPv4 Header) - Internet Header Length (IPv4 Header)

Μέγεθος δεδομένων Segment = Μέγεθος Segment - Data Offset (TCP Header)

1.22 40 bytes

1.23 Το πακέτο που στέλνει ο υπολογιστής μας (το SYN) έχει επιπρόσθετα 20 bytes TCP Options.

Άσκηση 2

2.1 `tcp && host 147.102.40.15`

Το port δεν τίθεται για τους λόγους που αναφέρονται στο ερώτημα 2.3.

2.2 21

2.3 Η θύρα δεδομένων FTP – η 20 – χρησιμοποιείται στην FTP Active Mode, όπου ζητάμε από τον εξυπηρετητή FTP, μέσω της Control σύνδεσης της θύρας 21, να ανοίξει εκείνος μια σύνδεση TCP με τον υπολογιστή μας στην θύρα FTP δεδομένων και στην σύνδεση που θα ανοίξει να μας στείλει τα δεδομένα.

Το `tnftp` – που είναι συνήθως ο default FTP client στο Linux – ως προεπιλογή χρησιμοποιεί την FTP Passive Mode (και πιο συγκεκριμένα την Extended Passive Mode που αποτελεί μικρή προσαρμογή για να δουλεύει και με IPv6), όπου ζητάμε από τον εξυπηρετητή FTP, μέσω της Control σύνδεσης της θύρας 21, να κάνει listen σε μια TCP θύρα, να μας στείλει τον αριθμό της θύρας μέσω της Control σύνδεσης και έτσι, τελικά, να ξεκινήσουμε εμείς την TCP σύνδεση δεδομένων FTP.

Ο λόγος που αυτό πολλές φορές είναι αναγκαίο είναι η ύπαρξη Τείχων Προστασίας που δεν επιτρέπουν σε υπολογιστές να ανοίξουν συνδέσεις προς έναν υπολογιστή. Ενδεικτικά αυτό το φαινόμενο συμβαίνει στα περισσότερα οικιακά δίκτυα, όπου έχουμε μετάφραση διευθύνσεων NAT και έτσι εξωτερικοί υπολογιστές δεν μπορούν να ξεκινήσουν συνδέσεις προς υπολογιστές του δικτύου.

(εκτός και αν ρυθμίσουμε Port Forwarding, όμως και πάλι τότε δεν θα μπορούσαν δύο διαφορετικοί υπολογιστές να έχουν FTP συνδέσεις δεδομένων με εξωτερικούς του δικτύου υπολογιστές).

Ενδιαφέρουσα συμπλήρωση: Δοκιμάστηκε FTP Active mode σε οικιακό δίκτυο και λειτούργησε! Η καταγραφή στο Wireshark έδειξε ότι ο εξωτερικός του δικτύου FTP εξυπηρετητής ξεκινούσε εκείνος σύνδεση προς τον υπολογιστή μας! Μάλιστα βλέπαμε τον FTP client να έχει αποστείλει εντολή `EPRT` για τοπική διεύθυνση (δηλαδή 192.168.XXX.XXX). Αυτό ακούγεται αδιανόητο, αφενός διότι βρισκόμαστε πίσω από NAT, αφετέρου διότι στον εξυπηρετητή εστάλη τοπική διεύθυνση. Κατόπιν μιας μικρής μελέτης, διαπιστώθηκε (και επιβεβαιώθηκε από τον Οδηγό Χρήσης της συσκευής) ότι ο router που μας παρέχει ο ISP (τουλάχιστον η συγκεκριμένη συσκευή που έχει σε εμάς δοθεί) υλοποιεί FTP Application-level Gateway! Με άλλα λόγια, ο router υλοποιεί deep packet inspection σε όλα τα πακέτα που διέρχονται από αυτό, ανιχνεύει τα FTP και τα τροποποιεί κατάλληλα ώστε ο FTP εξυπηρετητής αντί να λάβει την τοπική διεύθυνση και το port που έδωσε ο υπολογιστή μας, να λάβει την δημόσια διεύθυνση του router με κάποιο άλλο port της επιλογής του router, ο οποίος ρυθμίζει αυτόματα και το σχετικό port forwarding.

TL;DR Η θύρα δεδομένων FTP είναι η 20.

2.4 `tcp port 21`

2.5 Για την εγκατάσταση της TCP σύνδεσης απαιτούνται 3 τεμάχια TCP.

2.6 Ο υπολογιστή μας αποστέλλει τεμάχιο με SYN. Ο εξυπηρετητής απαντά με SYN,ACK, και έπειτα εμείς απαντάμε σε αυτό με ACK.

2.7 40 bytes για τα πρώτα 2 και 32 bytes για το 3ο.

2.8 0

Συμπλήρωση: Σύμφωνα με το RFC 793 – ίσχυε παλαιότερα – και σύμφωνα με το RFC 9293 – ισχύει σήμερα – τα τεμάχια αυτά θα μπορούσαν δυνητικά να μεταφέρουν και δεδομένα αν η εφαρμογή/υλοποίηση το επέλεγε.

2.9 12ms

2.10 Συμφωνεί.

2.11 Ο υπολογιστής μας θέτει: 1081079493 και ο FTP εξυπηρετητής θέτει: 3418983477

2.12 Σύμφωνα με την παράγραφο 3.1 (ορισμοί) και την παράγραφο 3.3 (Sequence Numbers) του RFC 793 (αντίστοιχα των παραγράφων 3.1 και 3.4 του RFC 9293 – ισχύει σήμερα), το τεμάχιο με το SYN όριζει τον Initial Sequence Number (ISN) και τα δεδομένα ξεκινάνε από το ISN+1. Με άλλα λόγια, το αρχικό τεμάχιο με το SYN, αν και δεν περιέχει δεδομένα, καταλαμβάνει μία ακριβώς τιμή των Sequence Numbers, τον Initial Sequence Number.

Έτσι, ο εξυπηρετητής όταν λάβει το τεμάχιο με το SYN, αναγνωρίζει ότι ελήφθει το SYN και έτσι στέλνει ένα πάκετο που έχει για Acknowledgement Number τον Sequence Number που έλαβε στο SYN + 1, δηλαδή τον Initial Sequence Number + 1 του υπολογιστή μας. (Ο Acknowledgement Number είναι ο μικρότερος Sequence Number που ακόμα ο παραλήπτης δεν έχει λάβει)

Συμπλήρωση: Η παράγραφος 3.3 του RFC 793 (αντίστοιχα η παράγραφος 3.4 του RFC 9293 – ισχύει σήμερα), ορίζει πως και το FIN θεωρείται πως συμβαίνει ακριβώς έναν Sequence Number μετά τον Sequence Number του τελευταίου byte δεδομένων.

Έτσι, όταν ο εξυπηρετητής FTP τερματίζει την TCP σύνδεση (σημειώνεται ότι στο FTP στέλνουμε την εντολή Goodbye στον εξυπηρετητή και εκείνος αναλαμβάνει να τερματίσει πρώτος την TCP σύνδεση), ο Sequence Number που αποστέλλει στο τελικό τεμάχιο με το FIN είναι:

$\text{Initial Sequence Number}_{\text{εξυπηρετητή}}$ + Μέγεθος δεδομένων που αποστάλθηκαν από τον εξυπηρετητή (σε bytes) + 1 (για το FIN)

2.13 Ο Sequence Number ένος τεμαχίου είναι ο Sequence Number του 1ου byte που υπάρχει στο τεμάχιο αυτό. Το τελευταίο τεμάχιο της τριπλής χειραψίας είναι απλά ένα τεμάχιο με την σημαία ACK, χωρίς να περιέχει δεδομένα. Παρόλο που δεν περιέχει δεδομένα, το τεμάχιο έχει Sequence Number τον Sequence Number του 1ου byte που θα μπορούσε να υπάρχει στο τεμάχιο, δηλαδή τον Initial Sequence Number του υπολογιστή μας + 1.

Ο αριθμός επιβεβαίωσης είναι ο Sequence Number του 1ου byte που αναμένουμε να λάβουμε και συνεπώς είναι ο Initial Sequence Number του FTP εξυπηρετητή + 1.

2.14 Βλ. ερώτημα 2.8

2.15 Σύμφωνα με το RFC 793 (το RFC 9293 σήμερα), η αριθμητική των Sequence Number γίνεται σε $\mod 2^{32}$. Έτσι η μέγιστη τιμή των Sequence Number και Acknowledgment Number είναι η $2^{32} - 1$.

Ωστόσο πρέπει να σημειωθεί ότι επειδή η αριθμητική γίνεται $\mod 2^{32}$, τα πρότυπα ορίζουν πως όταν γίνεται wrap των τιμών μετά το 2^{32} η σχέση των αριθμών πρέπει να διατηρείται. (το μέγιστο παράθυρο λήψης είναι 2^{30}, οπότε αυτό μπορεί να γίνει χωρίς αμφισημία στην σύγκριση των αριθμών, όπως περιγράφεται στην παράγραφο 2.3 του RFC 7323. Επιπρόσθετα για την προστασία από wrap των Sequence Numbers για χρόνο μέχρι τα πακέτα να εξαφανιστούν σίγουρα από το δίκτυο, δηλαδή να αποκτήσουν hop limit μηδέν (συμβατικά 120 sec), χρησιμοποιείται ο μηχανισμός PAWS που περιγράφεται στην παράγραφο 5 του RFC 7323).

Πρακτικά, συνεπώς δεν υπάρχει μέγιστη τιμή των Sequence Number και Acknowledgment Number αφού μπορούν να αυξάνονται συνέχεια, κάνοντας απλά cycle μετά το 2^{32}.

2.16 `ip.addr == 147.102.40.15 && (tcp.connection.syn || tcp.connection.synack || (tcp.dstport == 21 && tcp.ack == 1 && tcp.seq == 1))`

2.17 65536

2.18 65984

2.19 Window $\cdot\ 2^{\text{Window Scale Shift Count}}$, όπου κάθε πλευρά έχει δηλώσει το Window Scale της στα αρχικά τεμάχια <SYN> και <SYN,ACK>.

Συμπλήρωση: Σύμφωνα με το RFC 1323 το Window Scale ορίζεται αποκλειστικά στα αρχικά πακέτα SYN και SYN, ACK και αποτελεί offer, δηλαδή θα πρέπει το σχετικό TCP Option να τεθεί και από τις δύο πλευρές για να ισχύσει (μάλιστα θα πρέπει να έχει τέθει στο αρχικό τεμάχιο <SYN> – εξ ου και η ονομασία offer – προκειμένου να μπορέσει να τεθεί στο τεμάχιο <SYN,ACK>).

2.20 Ο υπολογιστής μας ανακοινώνει Shift Count = 2, δηλαδή Window Scale = $2^2 = 4$.

Ο FTP εξυπηρετητής ανακοινώνει Shift Count = 6, δηλαδή Window Scale = $2^6 = 64$.

2.21 Στο πάκετο με την σημαία SYN, στην TCP επικεφαλίδα, στα TCP Options, στο συνόλο πεδίων με Kind = 0x03 (σημαίνει Window Scale), στο πεδίο Shift Count. Συγκεκριμένα ο πολλαπλασιστής του Window ισούται με $2^{\text{Window Scale Shift Count}}$.

2.22 1460 bytes

2.23 Σύμφωνα με το RFC 9293 (παράγραφος 3.7.1) η τιμή MMS που στέλνει ένας host στο τεμάχιο SYN ισούται με:

effective MTU - fixed IP και TCP headers (δηλαδή χωρίς πιθανά options)

Ο fixed TCP Header είναι 20 bytes. Ο IP header, όμως, εξαρτάται αν χρησιμοποιούμε IPv4 ή IPv6.

Στην περίπτωση μας η διεπαφή δικτύου έχει MTU 1500 bytes και χρησιμοποιούμε IPv4 που έχει 20 bytes Fixed IPv4 Header.

Συνεπώς η τιμή MMS που πρέπει να αποστέλλει ο υπολογιστής μας είναι $1500 - 20 - 20 = 1460$ bytes, όπως και πράγματι αποστέλλει.

2.24 Στο πάκετο με την σημαία SYN, στην TCP επικεφαλίδα, στα TCP Options, στο συνόλο πεδίων με Kind = 0x02 (σημαίνει Maximum Segment Size), στο πεδίο MSS Value.

Συμπλήρωση: Το MSS Value έχει μεγέθος 2 bytes, που σημαίνει μέγιστο MSS=0xffff=65535 που είναι και το μέγιστο μέγεθος IP πακέτου. Ιστορικά, αυτό ήταν επαρκές. Ωστόσο, το IPv6 υποστηρίζει πακέτα και μεγαλύτερα από 65535 bytes (τα IPv6 Jumbograms μεγέθους ως 4 GiB – τα οποία βέβαια δεν υιοθετήθηκαν ευρέως). Για αυτό και στο RFC 2675 (και πλέον σήμερα και στο RFC 9293) ορίζεται πως αν ληφθεί στο MSS Value η τιμή 65535 αυτή θα πρέπει να αντιμετωπίζεται ως άπειρη και το MSS να βρίσκεται από Path MTU Discovery — η οποία, βέβαια, ούτως ή άλλως πρέπει να τρέξει αφού ακόμα και χωρίς IPv6 Jumbograms η MSS Value, ως MTU της διεπαφής του host, μπορεί να χρησιμοποιηθεί μόνο ως ένα άνω φράγμα του μεγέθους των πακέτων για την Path MTU Discovery, αφού, γενικά, διεπαφές ενδιάμεσων δρομολογητών μπορεί να έχουν μικρότερο MTU.

2.25 536

2.26 Σύμφωνα με το ερώτημα 2.23 είναι:

MSS Value = MTU - Fixed IP Header - Fixed TCP Header = 576 - 20 - 20 = 536 bytes

2.27 Το μέγιστο τεμάχιο που μπορούμε να στείλουμε (εννοώντας το τμήμα δεδομένων), αν δεν στείλουμε καθόλου TCP και IP options, είναι: 536 bytes.

Συμπλήρωση: Αν και αποτελεί (προφανή) λεπτομέρεια, το RFC 9293 (παράγραφος 3.7.1) σχολιάζει την μείωση του τεμαχίου αν σταλούν πρόσθετα TCP ή IP options και, συνολικά, ορίζει ότι το μέγιστο TCP τεμάχιο είναι:

`Eff.snd.MSS = min(SendMSS+20, MMS_S) - TCPhdrsize - IPoptionsize` όπου:

- `SendMSS` είναι η τιμή MSS που παρελήφθη από τον remote host. (ή 536 για το IPv4 ή 1220 για το IPv6, αν MSS Option δεν παρελήφθη)
- `MMS_S` είναι το μέγιστο μέγεθος μηνύματος που το στρώμα μεταφοράς TCP μπορεί να στείλει. (δηλαδή να παραδώσει προς το στρώμα δικτύου).
- `TCPhdrsize` είναι το μέγεθος του fixed TCP header και επιπρόσθετα όσων TCP Options πρόκειται να αποσταλλούν. Ισούται με 20 στην (σπάνια) περίπτωση που δεν αποστέλλονται TCP Options, αλλά μπορεί να είναι μεγαλύτερο αν πρόκειται να σταλούν TCP Options. (παρατηρούμε ότι στην παραπάνω σχέση τα 20 bytes είχαν προστεθεί στο `SendMSS` και τώρα αφαιρούντα μέσω του TCPhdrsize)
- `IPOptionsize` είναι το μέγεθος είναι IPv4 options ή IPv6 extension headers που τυχόν αποστέλλονται μαζί με το TCP τεμάχιο.

2.28 FIN

2.29 Ο FTP εξυπηρετητής

2.30 3, τα εξής:

1. Ο FTP εξυπηρετητής στέλνει τεμάχιο με την σημαία FIN.
2. Ο υπολογιστής μας απαντά κάνοντας Acknowledge τον Sequence Number του FIN (υπενθυμίζεται ότι το FIN καταλαμβάνει έναν Sequence Number μετά το πέρας των δεδομένων) και στο ίδιο τεμάχιο θέτει την σημαία FIN.
3. Ο FTP εξυπηρετητής απαντά κάνοντας Acknowledge τον Sequence Number του FIN που έστειλε ο υπολογιστής μας.

2.31 32 bytes

2.32 Δεν φέρουν δεδομένα.

Ωστόσο, γενικά, τα τεμάχια που φέρουν την σημαία FIN θα μπορούσαν να έχουν δεδομένα.Το τελευταίο τεμάχιο, πάντως, που στέλνεται από την υπολογιστή μας, και αναγνωρίζει το FIN του εξυπηρετητή, δεν θα μπορούσε να φέρει δεδομένα, αφού προηγούμενως ο υπολογιστής μας είχε στείλει FIN.

2.33 Μέγεθος TCP τεμαχίου: 20 bytes Fixed TCP Header + 12 bytes TCP Options + 0 bytes δεδομένα = 32 bytes

Μέγεθος IPv4 πακέτου: 20 bytes Fixed IPv4 Header + 0 bytes IP Options + 32 bytes TCP τεμάχιο = 52 bytes

Μέγεθος Ethenet πλαισίου: 14 bytes Ethernet Header + 52 bytes IPv4 πακέτο = 66 bytes

2.34 Ακριβώς ίδια με ερώτημα 2.33

2.35 Ο υπολογιστής μας μετέδωσε 51 bytes δεδομένων στην TCP σύνδεση FTP ελέγχου.

Ο FTP εξυπηρετητής μετέδωσε 587 bytes δεδομένων στην TCP σύνδεση FTP ελέγχου.

2.36 Στα τελευταία πακέτα της σύνδεσης που παραλαμβάνονται από την απέναντι πλευρά, βλέπω τον Relative Acknowledgment Number και αφαιρώ 2:

1 για τον Sequence Number του FIN και 1 διότι ο Acknowledgment Number είναι ο επόμενος Sequence Number που θα περίμενε ο host να παραλάβει (ο Sequence Number του αρχικού SYN είναι ο Relative Sequence Number 0)

2.37 `ip.addr == 147.102.40.15 && tcp.port == 20`

2.38 ο υπολογιστής μας ανακοινώνει MSS 1460 bytes και ο FTP εξυπηρετητής ανακοινώνει MSS 536 bytes.

2.39 Όμοια με ερώτημα 2.27 είναι 1460 bytes.

2.40 Το 1ο τεμάχιο αποστέλλεται από τον FTP εξυπηρετητή και έχει TCP Timestamp με βάση το ρολόι του εξυπηρετητή. Στον υπολογιστή μας είναι αδύνατον να συγχρονιστούμε με το ρολόι του εξυπηρετητή προκειμένου να ξέρουμε πόσος χρόνος έχει πέρασει από την στιγμή που το πακέτο εκπέμφθηκε (το σχετικό RFC 7323 δεν προβλέπει καν μονάδες για τα TCP Timestamps, απλά να είναι μια μη-φθίνουσα ακολουθία αριθμών)

Η καλύτερη εκτίμηση που μπορούμε να κάνουμε για το RTT από τα τεμάχια της τριπλής χειραψίας είναι μετρώντας τον χρόνο από τον χρόνο εκπομπής του τεμάχιου <SYN,ACK>, που έστειλε ο υπολογιστής μας (2ο της χειραψίας), ως τον χρόνο λήψης του τεμάχιου <ACK> που απάντησε σε αυτό ο FTP εξυπηρετητής (3ο της χειραψίας). Στο Wireshark αυτό μετράται 224 ms.

Συμπλήρωση: Προκειμένου ο αποστολέας να μην χρειάζεται να θυμάται χρόνους εκπομπής, υπάρχει το TCP Option `Timestamp`, στο οποίο ο αποστέλλεας θέτει στο πεδίο `TSval` τον χρόνο εκπομπής (με βάση το δικό του ρολόι). Ο παραλήπτης υποχρεούται να θέσει στο τεμάχιο ACK στο πεδίο `TSecr` (του TCP Option `Timestamp`) τον αριθμό `TSval` που είχε λάβει στο τεμάχιο για το οποίο στέλνει τώρα ACK. Έτσι, ο host βλέποντας στα τεμάχια ACK που παραλαμβάνει την διαφορά μεταξύ `TSecr` και της τιμής του ρολογιού του την χρονική στιγμή της λήψης μπορεί να βρει το RTT.

2.41 Παρατηρήσαμε ότι το Linux χρησομοποιήσουμε TCP Selective Acknowledgments. Για τους σκοπούς της άσκησης απενεργοποιήθηκαν προσωρινά τα TCP Selective Acknowledgments με την εντολή:

```
sysctl net.ipv4.tcp_sack=0
```

Αφού έγινε αυτό, σε κάποια τεμάχια παρατηρούμε ότι στέλνει σωρευτική επιβεβαίωση για πολλά τεμάχια που έλαβε μαζί.

2.42 Ο εξυπρετητής απέστειλε 118 τεμάχια με δεδομένα.

2.43 Ο υπολογιστής μας απέστειλε 94 τεμάχια με την σημαία ACK για τα δεδομένα που έλαβε.

2.44 65584

2.45 Κατά την τριπλή χειραψία, επειδή η σύνδεση ξεκινά από τον εξυπηρετητή FTP, ο υπολογιστής μας έχει αποστείλει μόνο το τεμάχιο <SYN,ACK>. Έτσι, η τιμή Window Scale ξεκινά να ισχύει από το ακριβώς επόμενο τεμάχιο, όπου και ορίζεται η πραγματική επιθυμητή τιμή του Window.

2.46 Ναι. Το Linux κάνει auto-tuning του TCP Receive Window. Παρατηρούμε ότι το μέγεθος συνεχώς αυξάνεται μέχρι τα 130KiB, με την μικρότερη τιμή του να είναι η αρχική, δηλαδή τα 65584 bytes.

2.47 Αν ο υπολογιστής μας ανακοίνωνε μηδενική τιμή για το παράθυρο, τότε ο FTP εξυπηρετητής θα σταματούσε να στέλνει δεδομένα.

Ωστόσο, σύμφωνα με το RFC 9293, ο FTP εξυπηρετητής ξεκινώντας από χρόνο Retransmission Timeout και αυξάνοντας εκθετικά το διάστημα επαναποστολών, θα έστελνε τεμάχια (τουλάχιστον ένα ή και μηδενικά δεδομένα), πρακτικά για να δει, μέσω του τεμαχίου ACK που θα απαντήσει ο υπολογιστής μας, αν άλλαξε το Receiving Window του υπολογιστή μας και, άρα, αν ο υπολογιστής μας μπορεί πλέον να αποδεχθεί δεδομένα.

2.48 Μέγεθος πλαισίου: 590 bytes

Μέγεθος Ethernet επικεφαλίδας: 14 bytes

Μέγεθος IPv4 επικεφαλίδας: 20 bytes

Μέγεθος TCP επικεφαλίδας: 32 bytes

2.49 Στο ερώτημα 2.39 καταγράψαμε την μέγιστη τιμή τεμαχίου που το TCP θα επέτρεπε να σταλεί, καθώς μεγαλύτερα από αυτά ούτως ή αλλως δεν θα γίνοταν δεκτά από τον υπολογιστή μας. Ο περιορισμός του ερωτήματος 2.39 τέθηκε από τον υπολογιστή μας, μέσω του Maximum Segment Size που ο υπολογιστής μας έστειλε

Ωστόσο, περιορισμοί προκύπτουν και από τα MTU των διεπαφών δικτύου των ενδιάμεσων δρομολογητών, αλλά και φυσικά από την MTU της διεπαφής δικτύου του FTP εξυπηρετητή. Μάλιστα, το RFC 9293, όπως είδαμε στο ερώτημα 2.27, προβλέπει ρητά την μείωση του μέγιστου τεμαχίου λόγω της MTU της διεπαφής δίκτου που πρόκειται να στείλει το τεμάχιο, ενώ επίσης, αναφέρει και πως πρέπει να γίνει και διαδικασία Path MTU Discovery για να βρεθεί η MTU που θα επιτρέψει στο τεμάχιο να διέλθει από όλους τους ενδιάμεσους δρομολογητές.

Συνεπώς, ο εξυπηρετητής FTP αναγκάστηκε να μειώσει το μέγεθος των τεμαχίων που στέλνει παρακάτω από το περιορισμό που ο υπολογιστής μας έθεσε (τα 1460 bytes του ερωτήματος 2.39), είτε επειδή περιορίστηκε από την MTU της διεπαφής δικτύου του είτε επειδή εκτέλεσε μια διαδικασία Path MTU Discovery, η οποία βρήκε ότι η διαδρομή από τον εξυπηρετητή μέχρι τον υπολογιστή μας αποδέχεται MTU 576 bytes.

2.50 Η υπηρεσία που προσφέρει το TCP στα ανώτερα στρώματα είναι αποστολή/παραλαβή ενός byte stream, χωρίς να ορίζονται από το ανώτερο στρώμα σαφή όρια μηνυμάτος. Ο διαχωρισμός σε τεμάχια υλοποιείται από το στρώμα μεταφοράς TCP και είναι διαφανής στα ανώτερα στρώματα. Κατ' επέκταση, αν ο εξυπηρετητής έπρεπε να στείλει δεδομένα μεγαλύτερης τιμής του 2.39, απλά το στρώμα TCP θα τα είχε σπάσει σε περισσότερα του ενός τεμάχια (όπως ήδη κάνει όταν ο FTP εξυπηρετητής παραδίδει το αρχείο PCATTCP.exe μεγέθους 60 KiB στο στρώμα μεταφοράς).

Το RFC 879, που αναφέρεται στο ερώτημα, σχολιάζει περιπτώσεις όπου ένας host πρέπει να δημιουργήσει ένα και μόνο ένα πακέτο IP που να έχει συγκεκριμένο μέγεθος (επειδή το στρώμα πάνω από το στρώμα δικτύου δεν υποστηρίζει εκείνο θρυμματισμό). Το RFC 879 αναφέρει ως χαρακτηριστικό παράδειγμα την αποστολή ICMP Echo Reply που είναι ένα και μόνο IP πακέτο, αλλά πρέπει οπωσδήποτε να περιέχει όλα τα δεδομένα που περιέχει το αντίστοιχο ICMP Request. Σε τέτοιες περιπτώσεις, μπορεί ο αποστολέας, ήδη πριν παραδώσει στο πρώτο στρώμα ζεύξης (που ο ίδιος είναι συνδεδμεένος), να εκτελέσει IP Framgentation, και έπειτα να μεταδώσει τα IP Fragments που θα προκύψουν.

2.51 Ο FTP εξυπηρετητής μετέδωσε 61440 bytes και ο υπολογιστής μας μετέδωσε 0 bytes.

2.52 $\frac{61.44 \text{ kbyte}}{3.373 \text{ sec}} = 18.22$ kbyte/sec

2.53 Ναι. Το Wireshark τις σημειώνει ως `TCP Retransmission`.

Συμπλήρωση: Με μια σύντομη, πρόχειρη μελέτη (σε fora) αυτό το κάνει παρατηρώντας αν ένα τεμάχιο φτάνει με διαφορετική σειρά στους Sequence Numbers από τα τεμάχια που είναι χρονικά πριν από αυτό. Αν η χρονική διαφορά είναι μικρή – μάλλον κάτω από 3 ms – το θεωρεί Out Of Order, ενώ αν είναι μεγάλη – μάλλον πάνω από 3 ms – το θεωρεί Retransmission)

Άσκηση 3

3.1 `tcp.port == 20`

3.2 94.65.141.44

3.3 14.5 ms

3.4 Παρατηρώ ότι ο αποστολέας στέλνει αρχικά 4 τεμάχια, και μετά, ανά περίπου χρόνο RTT, στέλνει συνεχώς περίπου διπλάσιο αριθμό τεμαχιών από προηγουμένως.

3.5 Στο πρώτο RTT έστειλε 5 τεμάχια, εκ των οποίων το 1ο τεμάχιο είναι το τεμάχιο <ACK> της τριπλής χειραψίας.

Όπως έχουμε βρει παραπάνω, ο εξυπηρετητής FTP – μάλλον – έχει SMSS=536 bytes. Σύμφωνα με την παράγραφο 3.1 του RFC 5681 πρέπει $IW \leq 4 \cdot 536$ bytes και το IW να μην αντιστοιχεί σε περισσότερα από 4 τεμάχια. Σημειώνεται ότι το 1ο από τα 5 τεμάχια που στάλθηκαν αντιστοιχεί στο acknowledgment του <SYN,ACK>, οπότε δεν προσμετράται στο IW, σύμφωνα με την παράγραφο 3.1 του RFC 5681. Άρα, το πλήθος τεμαχίων στο πρώτο RTT είναι πράγματι σύμφωνο με όσα προβλέπονται στην παράγραφο 3.1 του RFC 5681.

3.6 6 τεμάχια στο 2ο RTT και 10 τεμάχια στο 3ο RTT.

3.7 1 τεμάχιο στο 1ο RTT, 2 στο 2ο και 3 στο 3ο.

Το Slow Start, όπως ορίζεται στην παράγραφο 3.1 του RFC 5681, για κάθε ACK που παραλαμβάνει αυξάνει το Congestion Window κατά `SMSS`, που σύμφωνα με τους ορισμούς του ίδιου RFC είναι το μέγιστο μέγεθος τεμαχίου που μπορεί να αποσταλλεί λόγω σωρευτικά όλων των περιορισμών (MSS που έστειλε ο παραλήπτης, MTU διεπαφής, Path MTU Discovery, κ.λπ). Έτσι, αν ο παραλήπτης στείλει ισάριθμα ACK με τα τεμάχια που έλαβε, ο αποστολέας σε κάθε ριπή που θα στέλνει, θα διπλασιάζει το Congestion Window.

Σύμφωνα με την παράγραφο 4.2 του RFC 5681, ο παραλήπτης υποχρεούται να στέλνει τεμάχιο ACK το πολύ κάθε φορά που λαμβάνει $2 \cdot$ `RMSS`, όπου `RMSS` το MSS που είχε στείλει στον αποστολέα στην τριπλή χειραψία. (το RFC αυτό το αναφέρει ως 2 full-sized segments). Επίσης τα ACK θα πρέπει να μην αργούν παραπάνω από 500 ms από την στιγμή που παρελήφθη ένα τεμάχιο.

Λόγω των Delayed Acknowledgements αυτών, το Slow Start θα καταλήξει να μην διπλασιάζει το μέγεθος του Congestion Window σε κάθε ριπή που αποστέλλει. Για αυτό, το RFC 3465 στις παραγράφους 2.2 και 2.3, ορίζει πως ο αποστολέας μπορεί να αυξάνει το Congestion Window ως και $2 \cdot$ `SMSS` για κάθε τεμάχιο ACK που λαμβάνει.

Ωστόσο, παρατηρούμε πως ενώ ο παραλήπτης υποχρεούται να στείλει ACK κάθε διπλάσιο του MSS που έστειλε στην τριπλή χειραψία, ο αποστολέας μπορεί να αυξήσει το Congestion Window μόνο μέχρι το διπλάσιο του πραγματικού MSS που μπορεί να στείλει στον παραλήπτη, που φυσικά, μπορεί να είναι μικρότερο του MSS που μπορεί να δεχτεί ο παραλήπτης. Αυτή η "ασυμμετρία", συμβαίνει στην παρούσα περίπτωση, όπου ο παραλήπτης δήλωσε MSS=1452 bytes, όμως ο αποστολέας στέλνει με MSS=536 bytes. Αυτό, όπως είναι αναμενόμενο, οδηγεί σε ρυθμό αύξησης του Congestion Window μικρότερο από διπλασιασμό ανά ριπή.

Στο 6ο RTT αποστέλλονται 45 τεμάχια. Είναι: $4 \cdot \alpha^{6-1} = 45 \iff \alpha = 1.6$, δηλαδή σε κάθε ριπή το Congestion Window αυξάνεται 1.6 φορές αντί 2 φορών.

Θεωρητικά αν ACK στέλνεται ανά $2 \cdot 1452 = 2904$ bytes, ενώ για κάθε ACK το Congestion Window αυξάνεται $2 \cdot 536$ bytes, τότε σε μια ριπή μεγέθους `CWND` αποστέλλονται $\frac{\texttt{CWND}}{2 \cdot 1452}$ πλήθος τεμαχίων ACK, οπότε το νέο Congestion Window θα γίνει $\texttt{CWND} + \frac{2 \cdot 536}{2 \cdot 1452} \cdot \texttt{CWND} = 1.37 \cdot \texttt{CWND}$. Παρατηρήθηκε μεγαλύτερος ρυθμός αύξησης όμως (=1.6 φορές). Πράγματι, παρατηρώντας τα τεμάχια ACK στέλνονται ανά πλήθος από Sequence Numbers μη σταθέρο, μικρότερο, όμως, πάντα από $2 \cdot 1452 = 2904$ bytes.

Σε κάθε περίπτωση, παρατηρούμε ότι, τα Delayed Acknowledgments επιβραδύνουν κάπως το Slow Start. Ωστόσο, με μια προσεκτική σκέψη, παρατηρούμε ότι το πλήθος ριπών που απαιτείται για να φτάσουμε μια ορισμένη τιμή `CWND` με ορισμένο ρυθμό αύξησης α φορές/ριπή είναι $\log_\alpha \frac{\texttt{CWND}}{4}$. Όμως: $\log_\alpha x = \frac{\log_2 x}{\log_2 \alpha}$, οπότε αν $\alpha = (1-\kappa) \cdot 2$, τότε $\log_\alpha x = \frac{1}{1+\log_2(1-\kappa)} \cdot \log_2 x = (1 + \kappa + O(\kappa^2)) \log_2 x$, οπότε μια γραμμική μείωση του ρυθμού αύξησης, επιφέρει περίπου την ίδια γραμμική αύξηση του χρόνου επίτεξης της τιμής `CWND`, που είναι μάλλον ανεκτό.

3.8 Παρατηρούμε ότι το αρχικό πλήθος τεμαχίων που αποστέλλεται είναι 10, αντί για 4 που συνέβαινε πριν. Αυτό, πράγματι, επιτρέπεται από την παράγραφο 2 του RFC 6928 που ορίζει ότι το `IW` μπορεί να είναι το πολύ:

$$\min\left(10 \cdot MSS, \max\left(2 \cdot MSS, 14600\right)\right)$$

Στην περίπτωση μας αυτό καταλήγει $\min\left(10 \cdot 536, \max\left(2 \cdot 536, 14600\right)\right) = 10 \cdot 536$, δηλαδή 10 τεμάχια, όπως και πράγματι επιλέγει ο εξυπηρετητής να θέσει αρχικά το Congestion Window.

Άσκηση 4

4.1 `udp`

4.2

1. Source Port: 2 bytes
2. Destination Port: 2 bytes
3. Length: 2 bytes
4. Checksum: 2 bytes

4.3 8 bytes

4.4 UDP Size = IPv4 Total Length - IPv4 Header Length = 86 - 20 = 66 bytes

4.5 Το μέγεθος σε bytes του UDP δεδομενογράμματος συμπεριλαμβανομένων και των δεδομένων και της επικεφαλίδας

4.6 Σύμφωνα με το RFC 768, το ελάχιστο Length είναι 8 bytes (αντιστοιχεί μόνο σε επικεφαλίδα χωρίς δεδομένα).

4.7 Το ελάχιστο είναι: Ελάχιστη IPv4 επικεφαλίδα + Ελάχιστο UDP δεδομενόγραμμα = 20 + 8 = 28 bytes

Το Length έχει μέγεθος 2 bytes, οπότε δίνει μέγιστη τιμή 0xffff=65535. Το μέγιστο IPv4 πακέτο είναι 65535 bytes και μπορεί να προκύψει με 20 bytes IPv4 επικεφαλίδα, 8 bytes UDP επικεφαλίδα και 65507 bytes UDP δεδομένων.

4.8 Είναι 576 bytes - 8 bytes UDP επικεφαλίδα - 20 bytes IPv4 επικεφαλίδα = 548 bytes.

4.9 Δεν παρατηρήθηκαν.

4.10 `dns`

4.11 Στον υπολογιστή μου έχει ρυθμιστεί χειροκινήτα στο Google Public DNS, οπότε και μας απαντά ο 8.8.4.4.

4.12 Θύρα προέλευσης: 49982
Θύρα προορισμού: 53

4.13 Θύρα προέλευσης: 53
Θύρα προορισμού: 49982

4.14 Η 53.