אוניברסיטת
אריאל
בשומרון

פרויקט סיום

קורס תקשורת ומחשוב

תשפ״ב

מגישים:

חגי חן – 313414872

ניר גרון - 315874925

הסבר על הקוד:

ראשית, נצפה ונסביר את ההקלטת wireshark. רצף הפעולות שקורות בהקלטה הן:

1. חיבור של המשתמש לסרבר
2. המשתמש מכניס שם משתמש
3. הסרבר בודק האם השם אינו תפוס, במידה ולא מחזיר username accepted
4. לאחר מכן, המשתמש מכניס את הפקודה <download><bigfile.txt>, פקודה זו מבקשת מהשרת להוריד את הקובץ bigfile.txt
5. ההורדה מתחילה, לאחר מעבר של 50% מהקובץ, הסרבר מחכה לתשובה ממשתמש כדי להמשיך באמצעות הפקודה <proceed>, לאחר פקודה זו ההורדה ממשיכה.

בגדול, מה שנצפה לראות בהקלטה, בלוק של TCP שמהווה את הפעולות שקרו בסעיפים 1,2,3 ולאחר מכן בלוק של UDP שמהווה את הפעולות שקרו בסעיפים 4,5

נסביר את מהלך ההקלטה:

ראשית, לאחר חיבור של המשתמש לסרבר, המשתמש בוחר את השם שלו, במידה וניתן להשתמש בשם זה הסרבר מחזיר username accepted וכך בעצם הקליינט יודע האם הוא יכול להשתמש בשם זה. במידה ולא, תתקבל הודעת שגיאה, והשרת יתן למשתמש ניסיון חוזר.

```
 6 1.667936282   127.0.0.1   127.0.0.1   TCP   85 5002 → 39656
 7 1.667940767   127.0.0.1   127.0.0.1   TCP   68 39656 → 5002
 8 1.668057891   127.0.0.1   127.0.0.1   TCP   79 5002 → 39656
 9 1.668060653   127.0.0.1   127.0.0.1   TCP   68 39656 → 5002
10 8.570512223   127.0.0.1   127.0.0.1   TCP   91 39656 → 5002
▸ Frame 6: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface any, id 0
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 45 70 0c 40 00  40 06 cc a4 7f 00 00 01   E··Ep·@· @·......
0020  7f 00 00 01 13 8a 9a e8  cd fb c1 bd 12 3a bd 70   ........ ·····:·p
0030  80 18 02 00 fe 39 00 00  01 01 08 0a 93 26 7e f7   ·····9·· ·····&~·
0040  93 26 7e f7 75 73 65 72  6e 61 6d 65 20 61 63 63   ·&~·user name acc
0050  65 70 74 65 64                                     epted
```

לאחר מכן, השרת מחזיר חיווי למשתמש שהוא התחבר בהצלחה לשרת, <connected>

```
 8 1.668057891   127.0.0.1   127.0.0.1   TCP   79 5002 → 39656
 9 1.668060653   127.0.0.1   127.0.0.1   TCP   68 39656 → 5002
10 8.570512223   127.0.0.1   127.0.0.1   TCP   91 39656 → 5002
▸ Transmission Control Protocol, Src Port: 5002, Dst Port: 39656, Seq: 18, Ack: 2, Len: 11
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 3f 70 0d 40 00  40 06 cc a9 7f 00 00 01   E··?p·@· @·......
0020  7f 00 00 01 13 8a 9a e8  cd fb c1 ce 12 3a bd 70   ........ ·····:·p
0030  80 18 02 00 fe 33 00 00  01 01 08 0a 93 26 7e f7   ·····3·· ·····&~·
0040  93 26 7e f7 3c 63 6f 6e  6e 65 63 74 65 64 3e      ·&~·<con nected>
```

לאחר שהמשתמש מחובר לשרת, נרצה להוריד קובץ מהשרת, שם הקובץ הוא bigfile.txt, הודעה זו נשלחת מעל UDP.

```
12 8.570665663   127.0.0.1   127.0.0.1   UDP   69 57718 → 5002
13 8.571047541   127.0.0.1   127.0.0.1   TCP   83 5002 → 39656
14 8.571052322   127.0.0.1   127.0.0.1   TCP   68 39656 → 5002
15 8.607634458   127.0.0.1   127.0.0.1   UDP   46 53105 → 57718
16 8.607666977   127.0.0.1   127.0.0.1   UDP   49950 53105 → 57718
▸ User Datagram Protocol, Src Port: 57718, Dst Port: 5002
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 35 30 81 40 00  40 11 0c 35 7f 00 00 01   E··50·@· @··5····
0020  7f 00 00 01 e1 76 13 8a  00 21 fe 34 3c 64 6f 77   ·····v·· ·!·4<dow
0030  6e 6c 6f 61 64 3e 3c 62  69 67 66 69 6c 65 2e 74   nload><b igfile.t
0040  78 74 3e 7e 68                                     xt>~h
```

לאחר מכן, השרת מחזיר שהוא מתחיל לשלוח את ההודעה, כאשר המשתמש מקבל את הודעה זו, הוא מפעיל את הפונק' שמטפלת בקבלת הקובץ מהשרת.

```
13 8.571047541   127.0.0.1   127.0.0.1   TCP   83 5002 → 39656
14 8.571052322   127.0.0.1   127.0.0.1   TCP   68 39656 → 5002
15 8.607634458   127.0.0.1   127.0.0.1   UDP   46 53105 → 57718
16 8.607666977   127.0.0.1   127.0.0.1   UDP   49950 53105 → 57718
▸ Transmission Control Protocol, Src Port: 5002, Dst Port: 39656, Seq: 29, Ack: 25, Len: 15
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 43 70 0f 40 00  40 06 cc a3 7f 00 00 01   E··Cp·@· @·......
0020  7f 00 00 01 13 8a 9a e8  cd fb c1 d9 12 3a bd 87   ........ ·····:··
0030  80 18 02 00 fe 37 00 00  01 01 08 0a 93 26 99 ee   ·····7·· ·····&··
0040  93 26 99 ee 53 65 6e 64  69 6e 67 20 66 69 6c 65   ·&··Send ing file
0050  2e 2e 2e                                           ...
```

בתחילה, שולח השרת את מס׳ הפאקטות שהוא עתיד לשלוח, במקרה הנ״ל 26, ורק לאחר מכן מתחיל בשליחת הקבצים.

```
15 8.607634458   127.0.0.1   127.0.0.1   UDP   46 53105 → 57718
16 8.607666977   127.0.0.1   127.0.0.1   UDP   49950 53105 → 57718
17 8.607704528   127.0.0.1   127.0.0.1   UDP   49949 53105 → 57718
18 8.607726816   127.0.0.1   127.0.0.1   UDP   49948 53105 → 57718
19 8.607761451   127.0.0.1   127.0.0.1   UDP   49948 53105 → 57718
20 8.609588947   127.0.0.1   127.0.0.1   UDP   48 57718 → 53105
21 8.609801251   127.0.0.1   127.0.0.1   UDP   49949 53105 → 57718
▸ User Datagram Protocol, Src Port: 53105, Dst Port: 57718
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 1e 30 84 40 00  40 11 0c 49 7f 00 00 01   E···0·@· @··I····
0020  7f 00 00 01 cf 71 e1 76  00 0a fe 1d 32 36         ·····q·v ····26
```

לאחר שליחה של מס׳ הפקאטות, מתחילה בעצם שליחת המידע, תוך החזרה של ACK׳s מהקליינט

```
18 8.607726816   127.0.0.1   127.0.0.1   UDP   49948 53105 → 57718 Len=49904
19 8.607761451   127.0.0.1   127.0.0.1   UDP   49948 53105 → 57718 Len=49904
20 8.609588947   127.0.0.1   127.0.0.1   UDP   48 57718 → 53105 Len=4
▸ Frame 18: 49948 bytes on wire (399584 bits), 49948 bytes captured (399584 bits) on interface any, id
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 c3 0c 30 87 40 00  40 11 49 57 7f 00 00 01   E···0·@· @·IW····
0020  7f 00 00 01 cf 71 e1 76  c2 f8 c1 0c 32 7e 30 7e   ·····q·v ····2~0~
0030  61 20 66 61 63 69 6c 69  73 69 20 6e 75 6c 6c 61   a facili si nulla
0040  6d 20 76 65 68 69 63 75  6c 61 20 69 70 73 75 6d   m vehicu la ipsum
0050  20 61 20 61 72 63 75 2e  20 41 74 20 76 6f 6c 75    a arcu.  At volu
0060  74 70 61 74 20 64 69 61  6d 20 75 74 20 76 65 6e   tpat dia m ut ven
0070  65 6e 61 74 69 73 20 74  65 6c 6c 75 73 2e 20 43   enatis t ellus. C
0080  6f 6e 76 61 6c 6c 69 73  20 61 20 63 72 61 73 20   onvallis  a cras
0090  73 65 6d 70 65 72 20 61  75 63 74 6f 72 20 6e 65   semper a uctor ne
00a0  71 75 65 20 76 69 74 61  65 2e 20 56 69 74 61 65   que vita e. Vitae
00b0  20 73 65 6d 70 65 72 20  71 75 69 73 20 6c 65 63    semper  quis lec
00c0  74 75 73 20 6e 75 6c 6c  61 2e 49 6e 20 66 65 72   tus null a.In fer
00d0  6d 65 6e 74 75 6d 20 65  74 20 73 6f 6c 6c 69 63   mentum e t sollic
00e0  69 74 75 64 69 6e 20 61  63 20 6f 72 63 69 20 70   itudin a c orci p
00f0  68 61 73 65 6c 6c 75 73  2e 20 4a 75 73 74 6f 20   hasellus . Justo
0100  64 6f 6e 65 63 20 65 6e  69 6d 20 64 69 61 6d 20   donec en im diam
0110  76 75 6c 70 75 74 61 74  65 20 75 74 20 70 68 61   vulputat e ut pha
0120  72 65 74 72 61 20 73 69  74 2e 20 4d 61 75 72 69   retra si t. Mauri
0130  73 20 76 69 74 61 65 20  75 6c 74 72 69 63 69 65   s vitae  ultricie
0140  73 20 6c 65 6f 20 69 6e  74 65 67 65 72 20 6d 61   s leo in teger ma
0150  6c 65 73 75 61 64 61 20  6e 75 6e 63 2e 20 45 67   lesuada  nunc. Eg
0160  65 74 20 65 67 65 73 74  61 73 20 70 75 72 75 73   et egest as purus
0170  20 76 69 76 65 72 72 61  20 61 63 63 75 6d 73 61    viverra  accumsa
0180  6e 2e 20 50 65 6c 6c 65  6e 74 65 73 71 75 65 20   n. Pelle ntesque
0190  70 75 6c 76 69 6e 61 72  20 70 65 6c 6c 65 6e 74   pulvinar  pellent
01a0  65 73 71 75 65 20 68 61  62 69 74 61 6e 74 20 6d   esque ha bitant m
01b0  6f 72 62 69 2e 20 41 6c  69 71 75 61 6d 20 73 65   orbi. Al iquam se
01c0  6d 20 65 74 20 74 6f 72  74 6f 72 20 63 6f 6e 73   m et tor tor cons
01d0  65 71 75 61 74 20 69 64  2e 20 43 75 6d 20 73 6f   equat id . Cum so
01e0  63 69 69 73 20 6e 61 74  6f 71 75 65 20 70 65 6e   ciis nat oque pen
01f0  61 74 69 62 75 73 20 65  74 2e 20 56 6f 6c 75 74   atibus e t. Volut
```

ACK:

```
20 8.609588947   127.0.0.1   127.0.0.1   UDP   48 57718 → 53105
21 8.609801251   127.0.0.1   127.0.0.1   UDP   49949 53105 → 57718
▸ Frame 20: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface any, id 0
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 20 30 89 40 00  40 11 0c 42 7f 00 00 01   E·· 0·@· @··B····
0020  7f 00 00 01 e1 76 cf 71  00 0c fe 1f 41 43 4b 30   ·····v·q ····ACK0
```

מהלך זה מתבצע למעשה עד לשליחה של 50% מהקובץ, כשאר נגיע לשליחה של 50% השרת יחכה להודעה מהמשתמש על מנת להמשיך בשליחה הקבצים. <proceed>

```
49 12.558718919  127.0.0.1  127.0.0.1  UDP  53 57718 → 53105
50 12.558800867  127.0.0.1  127.0.0.1  TCP  80 5002 → 39656 [
User Datagram Protocol, Src Port: 57718, Dst Port: 53105
0000  00 00 03 04 00 06 00 00  00 00 00 00 08 00 08 00  ........ ........
0010  45 00 00 25 32 76 40 00  40 11 0a 50 7f 00 00 01  E··%2v@· @··P····
0020  7f 00 00 01 e1 76 cf 71  00 11 fe 24 3c 70 72 6f  ····· v·q ···$<pro
0030  63 65 65 64 3e                                    ceed>
```

לאחר שהמשתמש ביקש מהשרת להמשיך, השרת מחזיר לו <proceeding> על מנת שהמשתמש ידע שהשרת קיבל את ההודעה והוא ממשיך, לאחר שהמשתמש מקבל את ההודעה הזו, הוא יודע להמשיך את פעולת קבלת הקובץ.

```
50 12.558800867  127.0.0.1  127.0.0.1  TCP  80 5002 → 39656
51 12.558805442  127.0.0.1  127.0.0.1  TCP  68 39656 → 5002
Transmission Control Protocol, Src Port: 5002, Dst Port: 39656, Seq: 44, Ack: 34, Len: 12
0000  00 00 03 04 00 06 00 00  00 00 00 00 68 00 08 00  ........ ····h···
0010  45 00 00 40 70 11 40 00  40 06 cc a4 7f 00 00 01  E··@p·@· @·******
0020  7f 00 00 01 13 8a 9a e8  cd fb c1 e8 12 3a bd 90  ........ ·····:··
0030  80 18 02 00 fe 34 00 00  01 01 08 0a 93 26 a9 82  ·····4·· ·····&··
0040  93 26 a9 82 3c 70 72 6f  63 65 65 64 69 6e 67 3e  ·&··<pro ceeding>
```

בשלב זה, ממשיכה ההורדה – שליחת הקבצים מצד השרת והחזרת ה ACK's מצד המשתמש.
בסוף קבלת כל המידע הדרוש, המשתמש מחזיר הודעת end לשרת, וכך השרת יודע שכל הקובץ התקבל במלואו.

```
97 12.591217906  127.0.0.1  127.0.0.1  UDP  47 57718 → 53105 Len=3
User Datagram Protocol, Src Port: 57718, Dst Port: 53105
0000  00 00 03 04 00 06 00 00  00 00 00 00 6c 6f 08 00  ........ ····lo··
0010  45 00 00 1f 32 a4 40 00  40 11 0a 28 7f 00 00 01  E···2·@· @··(····
0020  7f 00 00 01 e1 76 cf 71  00 0b fe 1e 65 6e 64     ·····v·q ····end
```

מבנה הקוד:

לאחר שעברנו על ההקלטה, נרצה להסביר באופן מפורט על האלגוריתם שלנו ומבנה המחלקות.

MyServer.py

מחלקה זו, כפי שניתן להבין משמה, מאגדת בתוכה את כל פעולות השרת.

השרת מחזיק יומן של המשתמשים עם הסוקטים שלהם, רשימה של הקבצים שניתן להוריד מהשרת, וסוקטים מסוג UDP ו TCP.

כדי להפעיל את השרת, יש לייצר אובייקט מסוג MyServer ולהפעיל את הפונק' accept client באמצעות אובייקט מסוג MyServer.

def accept_clients:

למעשה, פונק' מאפשרת קבלת משתמשים אל השרת. עבור כל משתמש שמתחבר היא בודקת האם השם שבחר פנוי ומחזירה שגיאה במידה והשם שגוי לאחר שהתקבל שם שאינו תפוס הפונק' מחזירה חיווי על התחברות בהצלחה. בנוסף, הפונק' יוצרת שני תהליכים שמאזינים לפקטות מסוג TCP ו UDP.

def handle_udp:

פונק' זו מופעלת ע"י תהליך מהפונק' accept_clients. היא למעשה מאזינה לפקטות מסוג UDP בתהליך נפרד ופועלת בהתאם למידע שקיבלה מהמשתמש. כאשר היא מקבלת הודעה לבקשה להורדה היא יוצרת סוקט חדש מסוג UDP ואותו שולחת לפונק' send_file. היא עושה זאת על מנת לאפשר הורדה של קבצים במקביל, כי על אותו פורט ניתן לעבור רק מול משתמש אחד בו זמנית. לאחר סיום ההורדה השרת סוגר את הסוקט.

def handle_client:

פונק' זו מופעלת ע"י תהליך מהפונק' accept_clients. היא למעשה מאזינה לפקטות מסוג TCP בתהליך נפרד ופועלת בהתאם למידע שקיבלה מהמשתמש. אחראית על כל העברת ההודעות הרגילה, בין משתמשים ובין השרת ללקוח. הודעות כמו בקשה לקבל את כל המשתמשים, הודעות, קבצים וכו'.

def send_file:

פונק' זו שולחת את הקובץ אל המשתמש שביקש את הקובץ הנ"ל. לאחר שהפונק' handle_udp קיבלה בקשה להורדה, היא מפעילה את הפונק send_file, פונק' זו בתחילה עושה split, כלומר לוקחת את הקובץ שהיא צריכה לשלוח ומפצלת אותו לפקטות באורך ה buffer. כשאר משורשר לכל פקטה גם מס', שבאמצעותו המשתמש ידע מה מס' הפקטה שהתקבלה וכך יוכל להחזיר את ה ACK המתאים. לאחר מכן, שולחת חיווי למשתמש שהיא מתחילה לשלוח את הקובץ ושולחת את גודל הקובץ. לאחר מכן היא מתחילה בשליחת המידע. לאחר שנשלח 50% מהקובץ ע"י קבלת הACK's מהמשתמש, השרת מחכה לבקשה מהמשתמש להמשיך, לאחר קבלת בקשה זו הוא גם מחזיר לו שהוא אכן ממשיך. השרת מפסיק לעבוד כאשר הוא מקבל הודעת end.

Client.py

מחלקה זו, כפי שניתן להבין משמה, מאגדת בתוכה את כל פעולות המשתמש.

כל אובייקט מסוג Client מחזיק בתוכו סוקט מסוג TCP ו UDP, שם משתמש ורשימה של ההודעות שקיבל.

מלבד האתחול של השדות הנ"ל, הקונסטרקטור קולט את השם משתמש ומוודא מול השרת שאכן ניתן להשתמש בשם זה. לאחר סיום האתחול, הקונסטרקטור מפעיל תהליך (thread) המקשיב להודעות מסוג TCP.

def listen_for_messages:

מאזין להודעות שמקבל מהשרת ומפעיל את הפונק' בהתאם. בנוסף, ממוסיף את ההודעות לרשימה שמאגדת בתוכה את כל ההודעות שקיבל המשתמש.

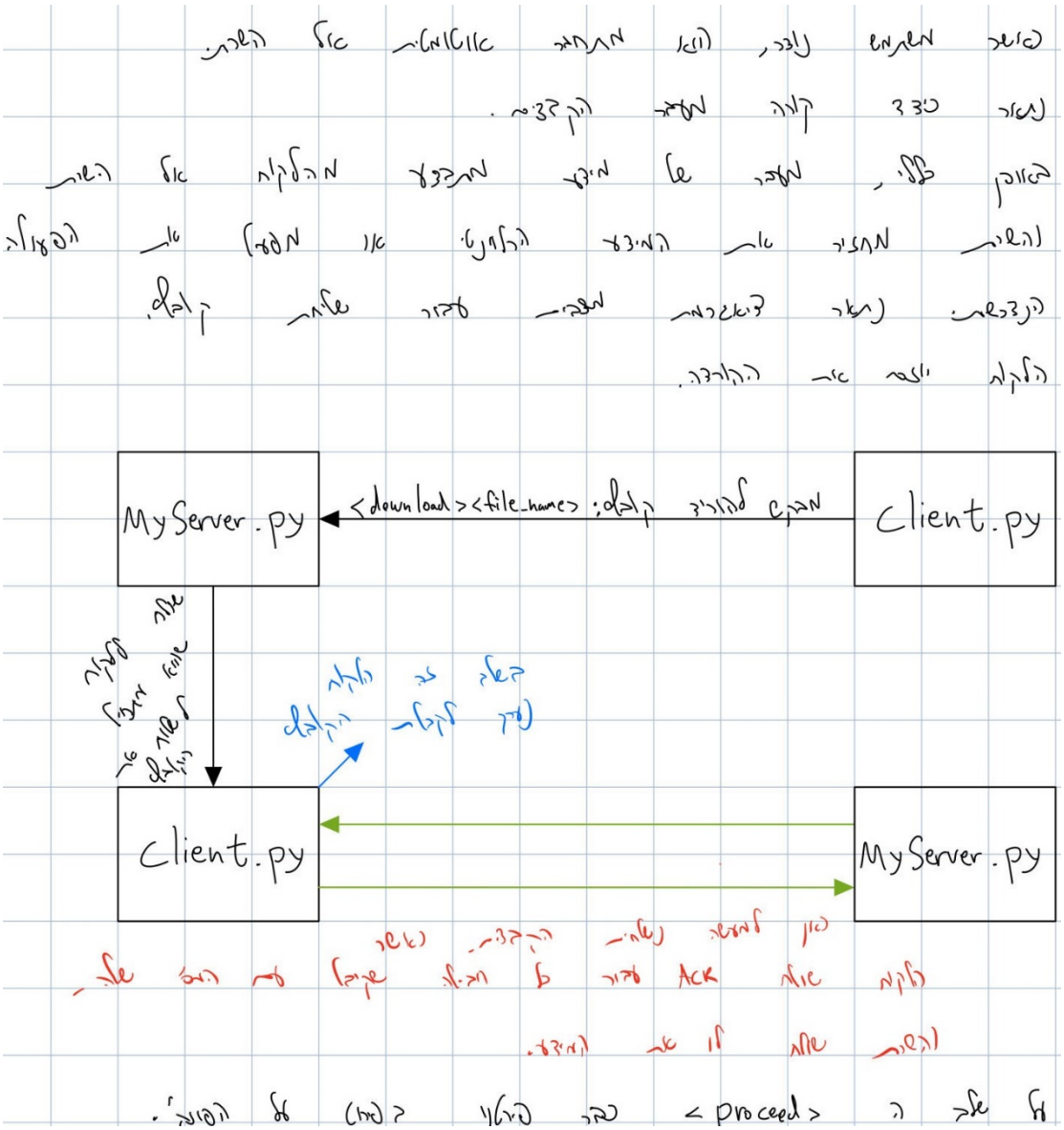
def send_message:

פונק' זו, קולטת מאזינה לקלט מהמשתמש ושולחת אותו אל השרת באמצעות קשר TCP.

def get_file:

פונק' זו מופעלת כאשר הפונק' listen_for_messages מקבל מהשרת כי הוא מתחיל בשליחה של קובץ, כאשר השרת שולח הודעה זו בעקבות בקשה של המשתמש (מעין 3-hand shake), פונק' זו מקבלת את המידע מהשרת, שומרת אותו ברשימה ומחזיקה ACK's בהתאם למס' הפאקטה.
כאשר מתקבל 50% מהמידע, השרת מחכה להודעה מהמשתמש להמשך ההורדה של הקובץ.
לאחר שהמשתמש מבקש את המשך השליחה, הפונק' ממשיכה בקבלת הקובץ. לאחר שקיבלה את כולו, מתחילה הפונק' בכתיבה של המידע על גבי קובץ.

דיאגרמת מצבים:



איבוד פאקטות:

כדי להתמודד עם איבוד פאקטות, כל חבילה שהלקוח מקבל, הוא מחזיר ACK עם מס׳ החבילה. (הלקוח יודע מה מספר החבילה כיוון שהיא משורשרת במידע שהוא מקבל מהשרת) כלומר, ACK0 אומר לשרת שהלקוח קיבל את הפאקטה הראשונה. וכן הלאה.
במידה והלקוח החזיר NACK זה אומר שהפקטה שהתקבלה לא הגיעה בסדר הנכון, כלומר בגלל שהפקטות נשלחות לפי סדר, ישנו סיכוי שאם קיבלנו פקטה אחת (למרות שנשלחה בשלב מאוחר יותר) לפני פקטה אחרת, אז יש לנו חבילה שאבדה ברשת. במצב כזה, הלקוח ישלח NACK עם מס׳ הפקטה שהייתה אמורה להגיע. לכן, עבור הודעת NACK שהשרת מקבל מהלקוח, השרת ישלח שוב את החבילה שהייתה אמורה להגיע ליעדה. (selective repeat)

בעיות latency:

תחילה נסביר כי בעיות latency הן בעיות הנובעות מכך שמרגע התחלת פעולה מסויימת, עד הרגע שהוא באמת מתחיל. במילים פשוטות, בעיות השהייה.

כאשר הלקוח רוצה לעשות פעולה מסויימת, הוא שולח הודעה לשרת, השרת מאשר קבלה, ורק לאחר אישור הקבלה הלקוח מתחיל בפעולה שרצה לעשות.
למשל, בשליחת קובץ, הלקוח מבקש להוריד קובץ מסויים, לאחר מעבר של 50% השרת מחכה להודעה מהלקוח על מנת להמשיך. הלקוח ממשיך בפעולת הקבלה רק לאחר קבלת המשך מהשרת. כך אנחנו יודעים שהשרת מוכן והוא ממשיך לשלוח הודעות. באופן דומה גם תחילת שליחת הקובץ מבצעת בדרך זו.

דוגמא נוספת, כאשר המשתמש רוצה להתנתק מהשרת, הוא שולח את ההודעה הזו לשרת והשרת יודע לנתק אותו. אך כדי שהוא אכן יקבל את ההודעה, עליו להמתין זמן מסויים כדי לקבל מהשרת את ההודעה שהוא התנתק. לכן השתמשנו בsleep קצר שיחכה לקבלת התשובה מהשרת.

חלק ג'

1. בהינתן שחיברנו מחשב חדש לרשת הוא צריך לקבל פרטי רשת ואת זה הוא עושה באמצעות פרוטוקול DHCP, הכרטיס שולח את ההודעה בbroadcast, כלומר כל הישויות ברשת יקבלו אותה, נקבל הודעה בחזרה בה יש כל פרטי הרשת כמו IP שרת DNS וכו'.
על מנת שההודעה תצליח להגיע אל שרת הDHCP על הלקוח להיות באותו Broadcast,
המחשב שלנו מחובר אל SWITCH אליו מחוברים מחשבים נוספים ברשת, וגם כמובן הRouter שלנו (נתב).
כעת, המחשב צריך לגלות מה הכתובת IP של הצד המקבל, על המחשב להשתמש בפרוטוקול DNS לתשאל את השרת הDNS מה הכתובת IP של הצד השני.
מכיוון שכתובת IP לא מספיקה אז גם צריך לגלות את הכתובת הפיזית של הנתב, תהליך זה מתבצע באמצעות פרוטוקול ARP, במידה ואין רשומה עבור הנתב ARP Cache של המחשב שלנו המחשב ישלח בתפוצה כללית (Broadcast) הודעה לכלל הרשת בחיפוש אחר הכתובת, פעולה זו נקראת ARP request, המחשב מקבל ARP reply , וכעת למחשב יש את כל המידע הדרוש על מנת לשלוח חבילה אל שרת הDNS.
switchה מקבל מהנתב את הARP reply ויש לו כבר טבלה עם מיפוי בין כתובות הMAC לבין הפורט הפיזי ובעצם הוא מבצע את הקבלה אל המחשב קצה.

ועכשיו באופן היותר מסודר לעין:
המחשב משיג כתובת IP – בהודעה יש כרטיס רשת שולח הודעת Discover DHCP, מקור – IP של המחשב יעד- כתובת IP של השרת DNS
Request ARP – המחשב שולח בBroadcast הודעה לכלל הרשת ובודק למי יש את הכתובת שאותה הוא מחפש
Reply ARP – החזרת הכתובת הפיזית המתאימה
התאמת כתובת ושיוך לפורט הפיזי אליו מחובר המחשב ע"י הSWITCH – כתובת הIP של המחשב שלנו והיעד הוא הDNS
גישה אל השרת – three way hand shake, מקור – מספר כלשהו מהמערכת הפעלה, יעד – פורט
בקשת הHTTP – הדפדפן פונה בבקשה לאותו אתר באמצעות הפרוטוקול HTTP
תשובה מהHTTP – במידה והוא מוכן להחזיר את העמוד שלו בלי עיכונים הוא יענה תשובה HTTP OK

2. CRC - Cyclic redundancy check

הוא קוד העוזר לאתר שגיאות בהעברת נתונים, השימוש בו הינו באופן מחזורי (ציקלי) ושיטתי לתיקון שגיאות.
בצד השולח, מחושב הCRC והוא נוסף למידע שאותו אנו מעבירים ובצד המקבל צריך לאשר באמצעות CRC שהמידע הועבר ללא שינויים.
כיצד הוא עובד – הקוד מבוסס על איזומורפיזם שבין וקטורים לפולינומים, כך שמסתכלים על כל וקטור באורך n כפולינום שמקדמיו הם קואורדינטות הווקטור.
השימוש שלו נוח בגלל קלות החישוב המתמטית שלו וביעילות שלו במציאת השגיאות.

3. HTTP – Hypertext Transfer Protocol הוא פרוטוקול התקשורת הנפוץ ביותר בעולמנו.
הגרסה 1.1 מתבססת ברובה על גרסה 1.0 וההבדלים בין גרסה זו לקודמתה הם שליטה טובה יותר במטמון, הוספת שיטות הבקשה options ו- trace , תוספות מתקדמות שמטרתן לייעל את אופן הפעולה של הפרוטוקול.
הגרסה האחרונה של HTTP שהיא הגרסה השנייה עבדה בצורה דומה ל1 ול1.1 בכך שהיא משתמשת בTCP, שהוא מאוד אמין אך גם מסורבל ואיטי ובגלל זה נוצר הUDP שהוא מהיר יותר באופן משמעותי אך פחות אמין ולכן אינו מתאים לאותם שימושים.
בפרוטוקול QUIC – Quick UDP Internet Protocol שפותח על ידי גוגל, לוקח את הUDP ומוסיף לה אמינות וסדר בניגוד לתקן TCP שמצריך הרבה מסרים בין המשתמש לשרת.

פרוטוקול QUIC עובד בצורה זהה לUDP כשהוא זקוק למספר מסרים נמוך באופן ניכר, מה שעוזר לנו לעבוד בצורה מהירה יותר.
4.מספרי PORT זו הדרך לזהות תהליך ספציפי (אפליקציה וכו') שאליו יש להעביר את הודעת האינטרנט או רשת כלשהי אחרת כשהיא מגיעה לשרת, כל ההתקנים המחוברים לרשת מגיעים עם יציאות PORT ייחודיים המוקצים להם ובכך ניתן לאתר את ההתקנים.

5. Subnet או בתרגום "רשת משנה" הוא כתובת נוספת המלווה כל מחשב (חוץ מכתובת הIP) ומציין למחשב איזה חלק מתוך כתובת הIP הוא ה network ID ואיזה חלק הוא ה- Host ID.
היא בנויה ממספר סיביות בכתובת הIP המשמשות לקביעת כתובת הרשת,
עלינו להגדיר subnet בכל מחשב ברשת הip/tcp. אם לא נעשה זאת, אותו מחשב לא יוכל לתקשר בפרוטוקול זה.

6. כתובת MAC – Media Access Control Address הינה כתובת המשמשת לזיהוי ייחודי של רכיבי רשת.
למה אנחנו צריכים כתובת MAC כשיש לנו כתובת IP? כתובת הMAC הינה כתובת העוזרת לנו למצוא את הרכיב בתוך הLAN בשכבת האינטרנט, לעומת הכתובת IP העובדת רק בשכבת הIP.

7. האמצעים ראוטר סוויץ' שני דרכי תקשורת
נשתמש בראוטר כדי לתקשר עם רשת חיצונית מאיתנו (אינטרנט), על מנת להמיר את הכתובת הפנימית לחיצונית, וכאשר נרצה להשתמש ברשת פנימית נשתמש בסוויץ', התקשורת תהיה מהירה יותר כי אין צורך להמיר כתובות.

8. בIPv4 יש 32 סיביות, כך שהיא מאפשרת באופן תאורטי 4,294,967,296 כתובות שונות, אך כאשר נגמרים כל האופציות עוברים לכתובות של IPv6 שמורכבת מ64 סיביות המשמשות לזיהוי תת הרשת ו64 סיביות המשמשות כמזהה ממשק ובכך יש לנו מספר הרבה יותר גדול של אופציות

9. e) 3c לומד על תת הנתב x בפרוטוקול eBGP
f) 3a לומד על תת הנתב x בפרוטוקול iBGP
g) 1c לומד על תת הנתב x בפרוטוקול eBGP
h) 2c לומד על תת הנתב x בפרוטוקול OSPF