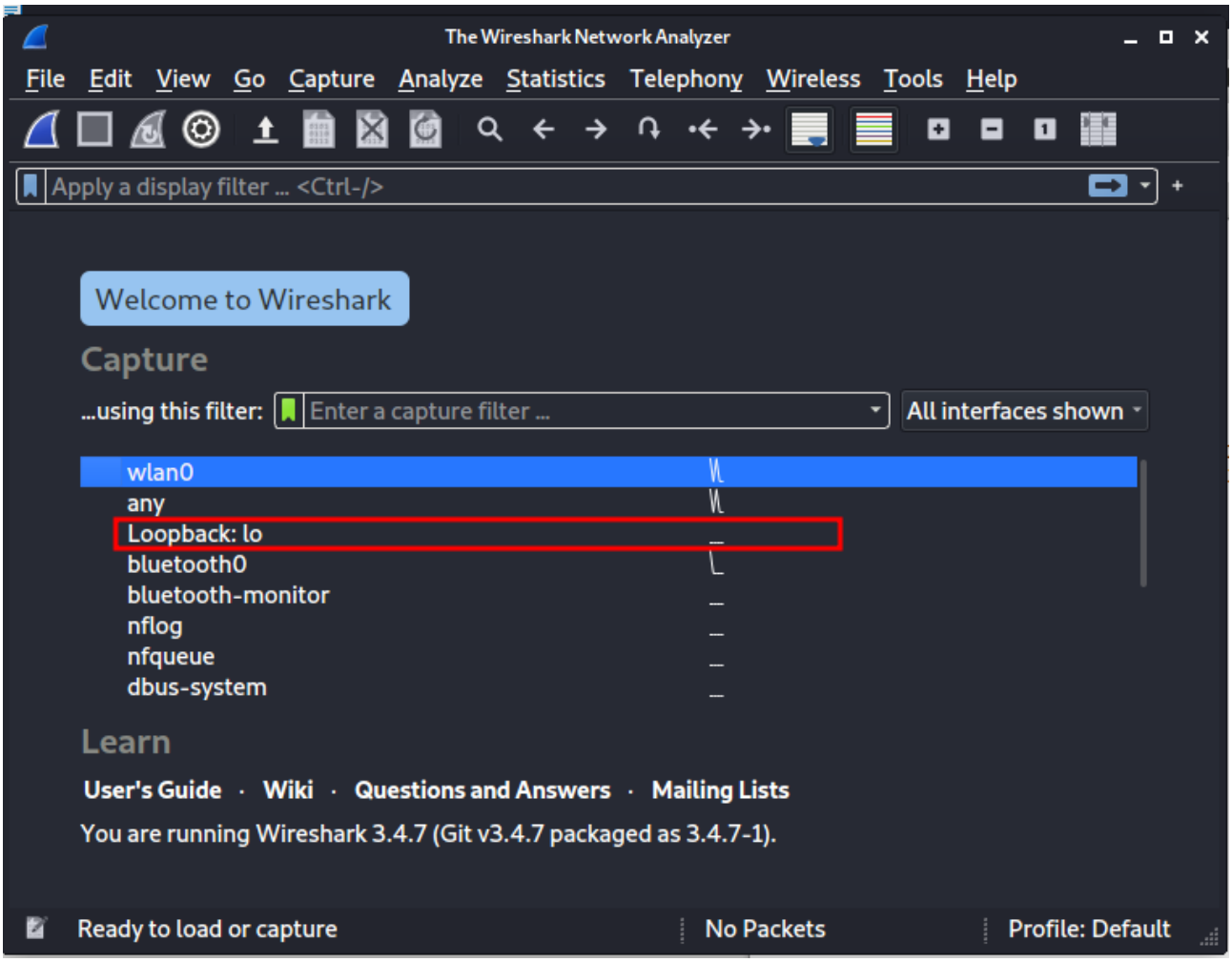
עבודה להגשה רשתות: עבודה מספר :1
מגיש: איתי אטליס – 209041474.

דגש: אני ובן עבדנו ביחד ולקראת הסוף החלטנו להתפצל בתרגיל זה, על כן ותיתכן לוגיקה דומה בתרגיל, אומנם המימוש עצמו שונה לחלוטין.

חלק א:
קיימות שתי שיטות לסנן את המידע המקרה שלנו, אני הלכתי על הדרך הפשוטה יותר .
בעת הפעלת התוכנה סימנתי כי ארצה להציג רק תעבורה שעוברת דרך הלופ-באק.



שיטה שניה:

נסמן בפילטר להציג רק תעבורה מכתובת הלופ-באק עם סינון עבור פרוטוקול יו די פי בלבד.

<u>שימוש בפורט:</u>
פרוטוקול יודיפי משתמש בפורט בכדי לדעת לאיזו אפליקציה עליו להעביר את המידע שקיבל .

הפורטים מוגדרים בשכבה 4/5 (שכבת התעבורה) עבור הפרוטקולים $(TCP\backslash UDP)$ שכבת ה session (האפליקציה במודל 5 השכבות) משתמשת בפורט בכדי ליצור סוקטים חדשים לתקשורת בין מכשירים שונים.

תחילה השרת יגדיר סוקט חדש המשתמש בפורט (12346 במקרה שלנו)

```
s.bind(('', 12346))
```

הלקוח יגדיר סוקט חדש גם כן אך יאפשר (במרבית המקרים) למערכת ההפלעלה להקצות פורט עבורו.

לאחר מכן הלקוח ישלח את המידע לאייפי והפורט של השרת.

```
1 0.000000000   127.0.0.1   127.0.0.1   UDP   64 40231 → 12345 Len=22
2 0.000060242   127.0.0.1   127.0.0.1   UDP   64 12345 → 40231 Len=22

▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface lo, id 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 40231, Dst Port: 12345
▸ Data (22 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 32 23 c0 40 00 40 11  18 f9 7f 00 00 01 7f 00   ·2#·@·@· ........
0020  00 01 9d 27 30 39 00 1e  fe 31 49 74 61 79 20 45   ···'09·· ·1Itay E
0030  74 65 6c 69 73 3a 20 32  30 39 30 34 31 34 37 34   telis: 2 09041474
```

כאשר הסורס פורט הינו הפורט שהמערכת הפעלה הקצתה, והדסטנטין פורט הוא הפורט שהגדרנו עבור השרת.

השימוש בפורטים הוא בשכבת התעבורה.

כתובות:IP

בשני המקרים כתובת האייפי הינה כתובת הלופ-באק כתובת שמורה המשמשת את מערכת ההפעלה בכדי לשלוח תעבורה פנימית בתוך המכונה עצמה.

```
▾ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 50
    Identification: 0xdd22 (56610)
  ▸ Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x5f96 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
▸ User Datagram Protocol, Src Port: 52095, Dst Port: 12346
▸ Data (22 bytes)
```

במקרה בו הכתובות שונות נוכל לראות כי עבור הפקאטה שנשלחה מן הלקוח כתובת הסורס תהיה כתובת האייפי שלו וכתובת הדסטנטיון תהיה כתובת השרת.

הכתובת המופיעה בפקודת ה$ipconfig$:

```
Wireless LAN adapter Wi-Fi 2:

   Connection-specific DNS Suffix  . : lan
   Link-local IPv6 Address . . . . . : fe80::7087:ff14:f9a2:becc%2
   IPv4 Address. . . . . . . . . . . : 192.168.1.108
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

מדוע הכתובות שונות?:
כפי שהוסבר כתובת הלופ-באק הינה כתובת שמורה במערכת ההפעלה עבור תקשורת פנים (כתובות בטווח: – 127.0.0.0 127.255.255.255) לעומת זאת הכתובת המופיע כdefault gateway: הינה כתובת ההזדהות של המחשב ברשת (יותר נכון של הראוטר) כלומר כל חבילה שתישלח מרשת זו תשלח עם IP מזהה של הdefault gateway
לעומת זאת כתובת הIPv4 הינה כתובת האישית של המכשיר עצמו להזדהות ברשת.

חלק ב':

הסבר האלגוריתם:

בהתאם להגדרת התרגיל האלגוריתם מנסה להעביר מידע בגודל עד 100 ביטים אל שרת היעד כאשר עובר דרך נתב foo אשר מדמה התנהגות רשת.

האלגוריתם ממומש בצורה המזכירה Pipeline, כאשר שולח חבילות מגודל 97 ביטים עם תוספת של שלושה ביטים המסמנים את גודל החבילה הנשלחת, כמו כן האלגוריתם שולח את כלל החבילות ברציפות ושומר מערך של מידע אצל השרת ומערך ack אצל הלקוח.

מונחים:

- חבילה רלוונטית – תא במערך DATA אשר מכיל את הערך הנכון לשליחה בבתים, חבילה שאינה רלוונטית מכילה את התו $b'0'$ בלבד.

לקוח:

תחילה הלקוח מאותחל על ידי מחלקה המחזיקה בכל המידע הנחוץ ומסדרת אותו בצורה כזו שכל גישה מבחוץ שומרת על אינקפסולציה.

שלב האתחול:

```
# Constructor - initialize variables and split file into small chunks.
def __init__(self, file_name):
    try:
        data = open(file_name, "rb").read()
    except OSError:
        print("Cannot open or read file")
        s.close()
        sys.exit()
    # Make new array of bytes from current file, to each segment add 3-byte long sequence number.
    self.data_arr = [(data[i:i + MSS] + int((i / MSS)).to_bytes(3, 'little')) for i in range(0, len(data), MSS)]
    # Size of the array.
    self.arr_size_bytes = (len(self.data_arr)).to_bytes(3, 'little')
    # Counter to hold the amount of packages received.
    self.received_coutner = 0
```

הבנאי מקבל את שם הקובץ ומנסה לקרוא ממנו, לאחר מכן המידע מחולק למערך אשר בכל תא ישנם 97 בתים של מידע ועוד 3 בתים של גודל הקובץ.

שלב יצירת החיבור:

```
# Sync method will be in charge of synchronizing connection with the server.
def syn():
    syn_msg = SYN_MSG + data_toSend.get_size()
    s.sendto(syn_msg, (IP_ADDR, PORT))
    # 2 ways handshake as the server is not communicating back but only receives data and approves it.
    try:
        get_syn, addr = s.recvfrom(100)
        if get_syn != syn_msg:
            syn()
        else:
            send_pkgs()
    except socket.timeout:
        # in-case of drop re-send package.
        syn()
```

בשלב זה הלקוח מנסה לגשת לשרת בעזרת המידע המסופק משורת ההרצה, שולח הודעת SYN עם מספר החבילות שעל השרת לצפות וממתין למענה מן השרת, אם אינו מקבל מענה בזמן מתאים ($Default - Timeout = 6sec$) שולח חבילה זו בשנית וכך הלאה באופן רקורסיבי.
כאשר מקבל את חבילת הSYN מן השרת עובר לחלק שליחת החבילות.

שליחת החבילות:

```
# will be in-charge of sending the packages that has not received ack yet.
def send_pkgs():
    for pkg in data_toSend:
        s.sendto(pkg, ADDR)
    mark_acks()
```

בשלב זה נשלחות כלל החבילות הרלוונטיות בעזרת איטרטור הממומש במחלקה:

```
# Iterator made for giving the next package in line that did not received ack for.
def __iter__(self):
    self.iter = 0
    return self

def __next__(self):
    # While we didn't reach the end of the array look for a package that have not received ack for.
    while self.iter < len(self.data_arr):
        # if found return it.
        if self.data_arr[self.iter] != NULL:
            temp = self.iter
            self.iter += 1
            return self.data_arr[temp]
        self.iter += 1
    raise StopIteration
```

האיטרטור עובר על כלל התאים במערך ומחזיר רק את התאים אשר אינם שווים ל$b'0'$

<u>קבלת הack:</u>

```
# will be in-charge of checking which acks have been received and marking the packages for future sending.
def mark_acks():
    try:
        while True:
            data_ack, address = s.recvfrom(100)
            if data_ack == FIN_MSG:
                fin()
            if data_ack == SYN_MSG:
                syn()
            pkg_num = int.from_bytes(data_ack[-3:len(data_ack)], 'little')
            data_toSend.notify_ack(pkg_num)

    except socket.timeout:
        if not data_toSend.done_acking():
            send_pkgs()
```

בשלב זה התוכנה רצה עד לקבלת *timeout*, התוכנית מנסה לקחת מהבאפר את המידע שהתקבל ולעדכן את מערך המידע בהתאם, כל חבילה מפורקת ומתקבל המספר הסידורי של החבילה, בעזרת מספר זה מתעדכן הערך במערך המידע במיקום זה ל$b'0'$.

<u>סיום החיבור:</u>

```python
# fin method will be in-charge to finish comunication with the server.
def fin():
    try:
        s.sendto(FIN_MSG, (IP_ADDR, PORT))
        fin_ack, address = s.recvfrom(100)
        if fin_ack != FIN_MSG:
            fin()
        else:
            s.close()
            sys.exit()
    except socket.timeout:
        fin()
```

לאחר קבלת כלל הacks מהשרת הלקוח סוגר את החיבור והתוכנה נסגרת, בנוסף קיימת אפשרות כי השרת קיבל את כלל החבילות אך ackים הלכו לאיבוד בדרך ועל כן בסיום השרת מתחיל לשלוח הודעות FIN, כאשר הלקוח מקבל אותן הוא מנתק את החיבור.

השרת:

השרת ממתין לחיבור חדש, בעת קבלת החיבור נוצר אובייקט חדש המנהל את כלל המידע של השרת ושומר על אינקפסולציה, עם קבלת החיבור השרת ממתין לקבלת חבילות וכל חבילה המתקבלת השרת מניח במערך מיועד במקום המתאים, עבור כל חבילה השרת שולח הודעת ACK ללקוח, בסיום קבלת החבילות השרת שולח הודעות התנתקות ללקוח (FIN) מאפס את החיבור הקיים וממתין לחיבור חדש.

שלב האתחול:

```python
# THe following class will hold all server's information and will be in-charged of the data itself.
class ServerData:
    # Constructor - will initialize all variables needed.
    # @param buffer - the buffer size of files about to arrive.
    def __init__(self, buffer):
        # Make new array of bytes from current file, to each segment add 3-byte long sequence number.
        self.ack_array = [False] * buffer
        self.received_coutner = 0
        self.array_size = len(self.ack_array)
        self.data_arr = [b''] * buffer
```

בעת קבלת חיבור חדש השרת מקבל לבנאי את כמות החבילות המצופות, מאתחל מערך חדש המכיל חבילות אלו ומערך בוליאני נוסף האחראי לעקוב אחר החבילות שהתקבלו.

שלב יצירת החיבור:

```python
# syn function will resemble the sync method used by the TCP protocol to initialise the connection.
def syn(syn_msg, addr):
    s.sendto(syn_msg, addr)
    syn_msg, addr = s.recvfrom(MSS)
    # if the message received is not the first package, than resend the syn ack confirmation.
    if syn_msg == SYN_MSG:
        syn(syn_msg, client)
    else:
        return syn_msg
```

השרת מקבל הודעת SYN מהלקוח, מחזיר הודעת SYN ללקוח וממתין, אם קיבל SYN נוסף אזי החבילה האחרונה אבדה ולכן שולח SYN נוסף, במידה וקיבל את החבילה הראשונה הוא מתחיל בעיבוד המידע.

שלב קבלת המידע:

```python
# if sync with client made successfully move to accept all packages from client.
if SYN_FLG:
    # The next while loop will iterate until all packages received, meaning bool_arr is all true.
    while not server_data.is_all_received():
        data, client = s.recvfrom(MSS)
        recv_pkg(data)
    # all packages has been successfully accepted, we will print the data and finnish connection with client.
    print(server_data.get_printable_data())
    fin(client)
```

כאשר החיבור הוקם הflag בשם SYN_FLG מקבל את הערך *TRUE* והתנאי מתקיים, מכאן השרת נכנס ללואה הנגמרת רק כאשר המטודה $is\_all\_received$. מחזירה *TRUE* מטודה זו אחראית לספור האם הגיעו כלל החבילות הרלוונטיות, במידה והגיעו כלל החבילות השרת מדפיס את המידע שקיבל ועובר לשלב סיום החיבור.

שלב שליחת ה *ACK*:

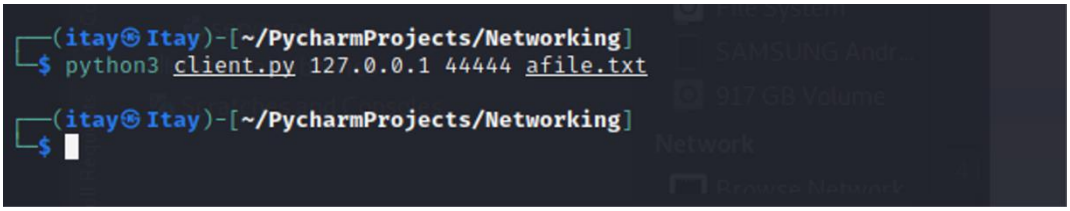
```python
# function will receive packages and insert true to the boolean array if value was not received before.
def recv_pkg(pkg):
    pkg_num = int.from_bytes(pkg[-3:len(pkg)], 'little')
    pkg_data = pkg[0:len(pkg) - 3]
    s.sendto(pkg, client)
    server_data.receive_ack(pkg_num, pkg_data)
```

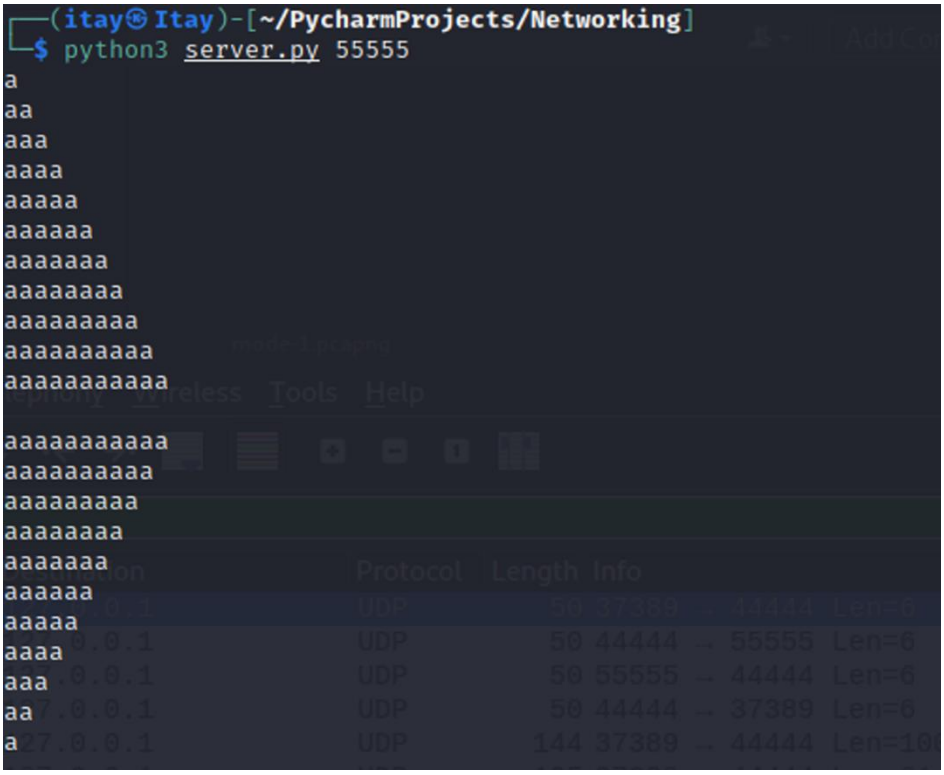
בשלב זה כל חבילה שמתקבלת נשלחת למטודה $receive\ ack$. מטודה זו אחראית להכניס את החבילה למאגר המידע במידה וטרם התקבלה בעבר, בכל מקרה עבור כל חבילה שמתקבלת מוחזר *ack*.

Wireshark:

ריצה ב $Mode = 1$:

ריצה פשוטה המדגימה את האלגוריתם מלעיל:





```
┌──(itay㉿Itay)-[~/PycharmProjects/Networking]
└─$ python3 foo.py 44444 127.0.0.1 55555 1
Playing nice
Phiiiii, no drop.... 66
Yay, no sleep.... 76 100
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 37389) to ('127.0.0.1', 55555
)
Phiiiii, no drop.... 16
Yay, no sleep.... 37 100
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 37389
)
Phiiiii, no drop.... 98
Yay, no sleep.... 12 100
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naa
aaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00' from ('127.0.0.1',
 37389) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 74
Yay, no sleep.... 35 100
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\n
a\n\x01\x00\x00' from ('127.0.0.1', 37389) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 68
Yay, no sleep.... 88 100
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naa
aaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00' from ('127.0.0.1',
 55555) to ('127.0.0.1', 37389)
Phiiiii, no drop.... 97
Yay, no sleep.... 48 100
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\n
a\n\x01\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 37389)
Phiiiii, no drop.... 30
Yay, no sleep.... 39 100
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 37389)
Phiiiii, no drop.... 70
Yay, no sleep.... 15 100
Forwarded b'FIN' from ('127.0.0.1', 37389) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 79
Yay, no sleep.... 27 100
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 37389)
```

אדום – ניתן לראות את הודעות תחילת התקשורת בין הלקוח לשרת ובין השרת ללקוח.

ירוק – ניתן לראות את המידע נשלח מן הלקוח לשרת עם סיומת של שלושה בתים המיצגים את המספר הסידורי של החבילה.

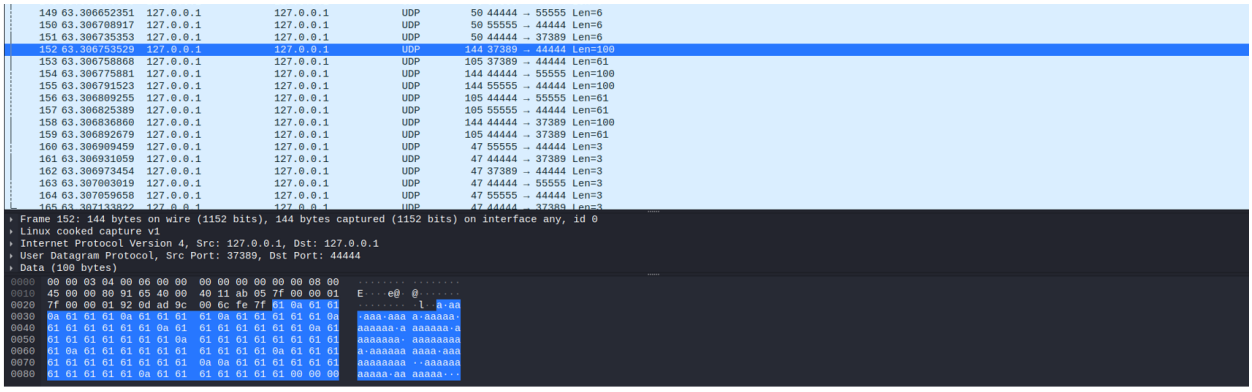
אפור – ניתן לראות את קבלת הACK מהשרת ללקוח

צהוב – הודעות סיום החיבור בין השרת ללקוח ($three\ handshake$)

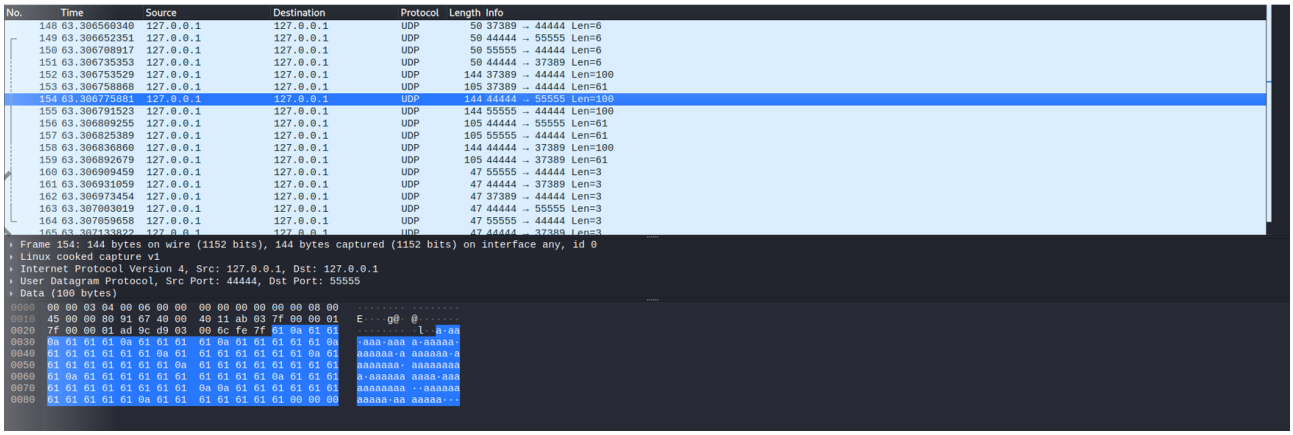
ip.addr == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
148	63.306560340	127.0.0.1	127.0.0.1	UDP	50	37389 → 44444 Len=6
149	63.306652351	127.0.0.1	127.0.0.1	UDP	50	44444 → 55555 Len=6
150	63.306708917	127.0.0.1	127.0.0.1	UDP	50	55555 → 44444 Len=6
151	63.306735353	127.0.0.1	127.0.0.1	UDP	50	44444 → 37389 Len=6
152	63.306753529	127.0.0.1	127.0.0.1	UDP	144	37389 → 44444 Len=100
153	63.306758868	127.0.0.1	127.0.0.1	UDP	105	37389 → 44444 Len=61
154	63.306775881	127.0.0.1	127.0.0.1	UDP	144	44444 → 55555 Len=100
155	63.306791523	127.0.0.1	127.0.0.1	UDP	144	55555 → 44444 Len=100
156	63.306809255	127.0.0.1	127.0.0.1	UDP	105	44444 → 55555 Len=61
157	63.306825389	127.0.0.1	127.0.0.1	UDP	105	55555 → 44444 Len=61
158	63.306836860	127.0.0.1	127.0.0.1	UDP	144	44444 → 37389 Len=100
159	63.306892679	127.0.0.1	127.0.0.1	UDP	105	44444 → 37389 Len=61
160	63.306909459	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
161	63.306931059	127.0.0.1	127.0.0.1	UDP	47	44444 → 37389 Len=3
162	63.306973454	127.0.0.1	127.0.0.1	UDP	47	37389 → 44444 Len=3
163	63.307003019	127.0.0.1	127.0.0.1	UDP	47	44444 → 55555 Len=3
164	63.307059658	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
165	63.307133822	127.0.0.1	127.0.0.1	UDP	47	44444 → 37389 Len=3

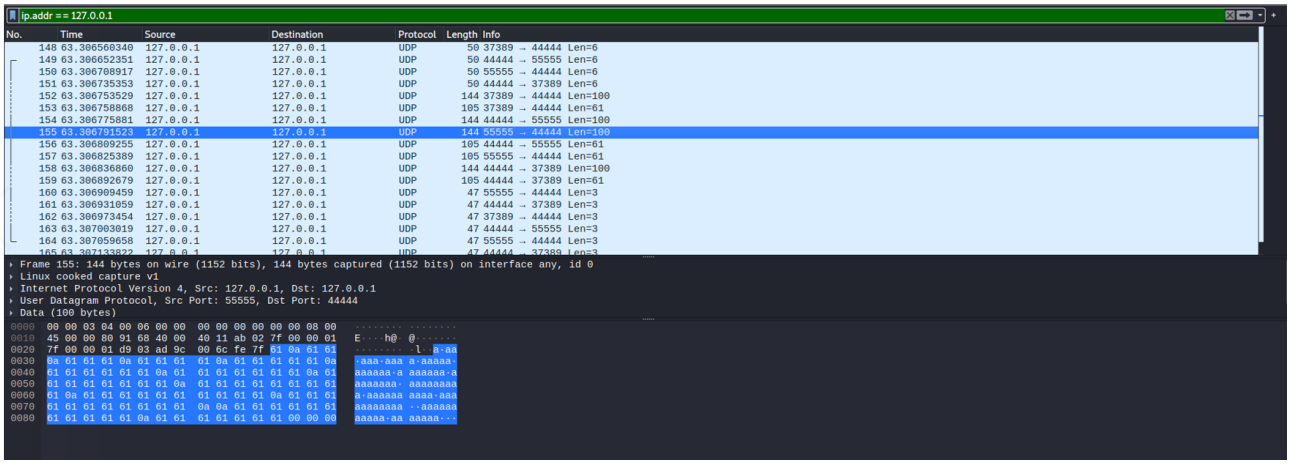
ניתן לראות את העברת ההודעת מהלקוח ל*FOO*:



ניתו לראות את העברת ההודעת מ*FOO* לשרת:



ניתן לראות את העברת ההודעה מהשרת ל *FOO*:



ריצה ב $Mode = 2$:

תמונת המצב של השרת והלקוח זהות ל – 1 ולכן מוצג רק *FOO*:

```
┌──(itay㉿Itay)-[~/PycharmProjects/Networking]
└─$ python3 foo.py 44444 127.0.0.1 55555 2
Dropping 52%
Dropped! randomly ...  62
Phiiiii, no drop.... 4
Yay, no sleep.... 29 100
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 53760) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 16
Yay, no sleep.... 87 100
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 53760)
Dropped! randomly ...  92
Dropped! randomly ...  60
Phiiiii, no drop.... 26
Yay, no sleep.... 45 100
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\n
aaaaaaa\x00\x00\x00' from ('127.0.0.1', 53760) to ('127.0.0.1', 55555)
Dropped! randomly ...  90
Phiiiii, no drop.... 15
Yay, no sleep.... 56 100
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\n
aaaaaaa\x00\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 53760)
Dropped! randomly ...  63
Phiiiii, no drop.... 46
Yay, no sleep.... 14 100
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 53
760) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 8
Yay, no sleep.... 34 100
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 55
555) to ('127.0.0.1', 53760)
Dropped! randomly ...  98
```

אדום – ניתן לראות את הודעות תחילת התקשורת בין הלקוח לשרת ובין השרת ללקוח.

ירוק – ניתן לראות את המידע נשלח מן הלקוח לשרת עם סיומת של שלושה בתים המיצגים את המספר הסידורי של החבילה.

אפור – ניתן לראות את קבלת ה*ACK* מהשרת ללקוח

ניתן לראות דבר מעניין, הלקוח שלח שתי חבילות מידע ואחת מהן נפלה בדרך, על כן בזמן זה השרת החזיר *ACK* על החבילה הראשונה ועל כן הלקוח היה צריך לשלוח את החבילה השניה בשנית ורק לאחר קבלת *ACK* עליה הוא מתנתק.

93	21.717414496	127.0.0.1	127.0.0.1	UDP	144	53760 → 44444 Len=100
94	21.717453739	127.0.0.1	127.0.0.1	UDP	105	53760 → 44444 Len=61
95	21.717648269	127.0.0.1	127.0.0.1	UDP	144	44444 → 55555 Len=100
96	21.717847056	127.0.0.1	127.0.0.1	UDP	144	55555 → 44444 Len=100

- מדוע אין הודעות *FIN*?: אין הודעות *FIN* כיוון שהלקוח הספיק לקבל *ACK* על כלל החבילות והודעת ה*FIN* מהשרת נפלה בדרך, על כן עם סיום קבלת ה*ACKS* הלקוח מתנתק, השרת מקבל *TIMEOUT* בהמתנה להודעה מהלקוח ועל כן מסיק כי הלקוח התנתק ומתנתק בעצמו.

ip.addr == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
86	11.705520403	127.0.0.1	127.0.0.1	UDP	50	53760 → 44444 Len=6
87	16.711042355	127.0.0.1	127.0.0.1	UDP	50	53760 → 44444 Len=6
88	16.711292114	127.0.0.1	127.0.0.1	UDP	50	44444 → 55555 Len=6
89	16.711481642	127.0.0.1	127.0.0.1	UDP	50	55555 → 44444 Len=6
90	16.711684566	127.0.0.1	127.0.0.1	UDP	50	44444 → 53760 Len=6
91	16.711847709	127.0.0.1	127.0.0.1	UDP	144	53760 → 44444 Len=100
92	16.711885173	127.0.0.1	127.0.0.1	UDP	105	53760 → 44444 Len=61
93	21.717414496	127.0.0.1	127.0.0.1	UDP	144	53760 → 44444 Len=100
94	21.717453739	127.0.0.1	127.0.0.1	UDP	105	53760 → 44444 Len=61
95	21.717648269	127.0.0.1	127.0.0.1	UDP	144	44444 → 55555 Len=100
96	21.717847056	127.0.0.1	127.0.0.1	UDP	144	55555 → 44444 Len=100
97	21.718041196	127.0.0.1	127.0.0.1	UDP	144	44444 → 53760 Len=100
102	26.723628497	127.0.0.1	127.0.0.1	UDP	105	53760 → 44444 Len=61
103	31.729174631	127.0.0.1	127.0.0.1	UDP	105	53760 → 44444 Len=61
104	31.729413096	127.0.0.1	127.0.0.1	UDP	105	44444 → 55555 Len=61
105	31.729523578	127.0.0.1	127.0.0.1	UDP	105	55555 → 44444 Len=61
106	31.729710865	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
107	31.729736912	127.0.0.1	127.0.0.1	UDP	105	44444 → 53760 Len=61

ריצה ב$Mode = 3$:

תמונת המצב של השרת והלקוח זהות ל – 1 ולכן מוצג רק *FOO*:

```
┌──(itay㉿Itay)-[~/PycharmProjects/Networking]
└─$ python3 foo.py 44444 127.0.0.1 55555 3
Delaying 88%
Phiiiii, no drop.... 81
Good night.... 55 12
Delaying! for 2.141 seconds ...
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 33859) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 32
Good night.... 95 12
Delaying! for 0.392 seconds ...
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 33859)
Phiiiii, no drop.... 97
Good night.... 66 12
Delaying! for 3.766 seconds ...
Phiiiii, no drop.... 73
Good night.... 68 12
Delaying! for 3.902 seconds ...
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00'
from ('127.0.0.1', 33859) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 7
Good night.... 96 12
Delaying! for 1.777 seconds ...
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 33859) to ('127.0.0.1',
 55555)
Phiiiii, no drop.... 83
Good night.... 65 12
Delaying! for 1.484 seconds ...
Phiiiii, no drop.... 19
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 33859)
Good night.... 75 12
Forwarded b'FIN' from ('127.0.0.1', 33859) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 25
Good night.... 50 12
Delaying! for 2.268 seconds ...
Forwarded b'aaa\naaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1',
 33859)
Phiiiii, no drop.... 72
Good night.... 16 12
Delaying! for 4.389 seconds ...
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00'
from ('127.0.0.1', 55555) to ('127.0.0.1', 33859)
Phiiiii, no drop.... 15
Good night.... 72 12
Delaying! for 2.125 seconds ...
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 33859)
Forwarded b'FIN' from ('127.0.0.1', 33859) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 20
```

<u>אדום</u> – ניתן לראות את הודעות תחילת התקשורת בין הלקוח לשרת ובין השרת ללקוח.

<u>ירוק</u> – ניתן לראות את המידע נשלח מן הלקוח לשרת עם סיומת של שלושה בתים המיצגים את המספר הסידורי של החבילה.

<u>אפור</u> – ניתן לראות את קבלת ה*ACK* מהשרת ללקוח

<u>צהוב</u> – הודעות סיום החיבור בין השרת ללקוח ($three\ handshake$)

דבר מעניין שניתן לראות הוא כי השרת סיים את קבלת הקבצים וכלל ה*ACK* התעכבו ועל כן החלו להשלח הודעות *FIN* טרם שליחת הודעות ה *ACK*, במקרה בו היה כמות מידע מסיבית יותר הלקוח היה מתנתק טרם קבלת כלל ה*ACK* בפקודת השרת.
ניתן לראות בתמונה הבאה כי אנו מקבלים הודעות כי הלקוח אינו זמין כאשר *FOO* מנסה להעביר אליו חבילות.

ip.addr == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
5	6.308946024	127.0.0.1	127.0.0.1	UDP	50	33859 → 44444 Len=6
6	8.452021701	127.0.0.1	127.0.0.1	UDP	50	44444 → 55555 Len=6
7	8.452165084	127.0.0.1	127.0.0.1	UDP	50	55555 → 44444 Len=6
8	8.845205973	127.0.0.1	127.0.0.1	UDP	50	44444 → 33859 Len=6
9	8.845384068	127.0.0.1	127.0.0.1	UDP	144	33859 → 44444 Len=100
10	8.845420005	127.0.0.1	127.0.0.1	UDP	105	33859 → 44444 Len=61
17	12.616184147	127.0.0.1	127.0.0.1	UDP	144	44444 → 55555 Len=100
18	12.616373922	127.0.0.1	127.0.0.1	UDP	144	55555 → 44444 Len=100
19	12.752547025	127.0.0.1	127.0.0.1	UDP	105	44444 → 55555 Len=61
20	12.752676311	127.0.0.1	127.0.0.1	UDP	105	55555 → 44444 Len=61
21	12.752913058	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
22	12.753551984	127.0.0.1	127.0.0.1	UDP	47	44444 → 33859 Len=3
23	12.753695488	127.0.0.1	127.0.0.1	UDP	47	33859 → 44444 Len=3
24	12.966707534	127.0.0.1	127.0.0.1	UDP	47	44444 → 55555 Len=3
25	12.966873175	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
26	14.238174837	127.0.0.1	127.0.0.1	UDP	105	44444 → 33859 Len=61
27	14.238347956	127.0.0.1	127.0.0.1	UDP	47	33859 → 44444 Len=3
28	14.396283089	127.0.0.1	127.0.0.1	UDP	144	44444 → 33859 Len=100
29	14.396444611	127.0.0.1	127.0.0.1	UDP	47	33859 → 44444 Len=3
30	15.238321054	127.0.0.1	127.0.0.1	UDP	47	44444 → 33859 Len=3
34	16.524330275	127.0.0.1	127.0.0.1	UDP	47	44444 → 55555 Len=3
35	16.524515895	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
36	18.632480029	127.0.0.1	127.0.0.1	UDP	47	44444 → 55555 Len=3
37	18.632614155	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
60	19.280314458	127.0.0.1	127.0.0.1	UDP	47	44444 → 33859 Len=3
61	19.280336322	127.0.0.1	127.0.0.1	ICMP	75	Destination unreachable (Port unreachable)
79	20.325483515	127.0.0.1	127.0.0.1	UDP	47	44444 → 33859 Len=3
80	20.325504375	127.0.0.1	127.0.0.1	ICMP	75	Destination unreachable (Port unreachable)

ריצה ב $Mode = 4$:

תמונת המצב של השרת והלקוח זהות ל – 1 ולכן מוצג רק *FOO*:

```
┌──(itay㉿Itay)-[~/PycharmProjects/Networking]
└─$ python3 foo.py 44444 127.0.0.1 55555 4
Dropping 44%
Delaying 12%
Dropped! randomly ...  97
Phiiiii, no drop.... 34
Yay, no sleep.... 58 88
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 54089) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 33
Yay, no sleep.... 31 88
Forwarded b'SYN\x02\x00\x00' from ('127.0.0.1', 55555) to ('127.0.0.1', 54089)
Dropped! randomly ...  78
Phiiiii, no drop.... 12
Good night.... 95 88
Delaying! for 2.166 seconds ...
Forwarded b'aaa\naaaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 54089) to ('127.0.0.1', 55555)
Dropped! randomly ...  74
Dropped! randomly ...  79
Dropped! randomly ...  95
Dropped! randomly ...  76
Dropped! randomly ...  64
Dropped! randomly ...  95
Dropped! randomly ...  88
Dropped! randomly ...  99
Dropped! randomly ...  68
Phiiiii, no drop.... 20
Yay, no sleep.... 53 88
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00' from ('127.0.0.1', 54089) to ('127.0.
0.1', 55555)
Phiiiii, no drop.... 3
Yay, no sleep.... 59 88
Forwarded b'aaa\naaaaaaaaaa\naaaaaaaa\naaaaaaa\naaaaaa\naaaaa\naaaa\naa\na\n\x01\x00\x00' from ('127.0.0.1', 54089) to ('127.0.0.1', 55555)
Phiiiii, no drop.... 41
Yay, no sleep.... 20 88
Forwarded b'a\naa\naaa\naaaa\naaaaa\naaaaaa\naaaaaaa\naaaaaaaa\naaaaaaaaa\naaaaaaaaaa\naaaaaaaaaaa\n\naaaaaaaaaaa\naaaaaaa\x00\x00\x00' from ('127.0.0.1', 55555) to ('127.0.
0.1', 54089)
Phiiiii, no drop.... 5
Good night.... 96 88
Delaying! for 1.191 seconds ...
Phiiiii, no drop.... 51
Yay, no sleep.... 36 88
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 54089)
Dropped! randomly ...  57
Forwarded b'FIN' from ('127.0.0.1', 55555) to ('127.0.0.1', 54089)
```

אדום – ניתן לראות את הודעות תחילת התקשורת בין הלקוח לשרת ובין השרת ללקוח.

ירוק – ניתן לראות את המידע נשלח מן הלקוח לשרת עם סיומת של שלושה בתים המיצגים את המספר הסידורי של החבילה.

אפור – ניתן לראות את קבלת ה*ACK* מהשרת ללקוח

צהוב – הודעות סיום החיבור בין השרת ללקוח ($three\ handshake$)

כאן ניתן לראות בבירור כי ה*ACK* על אחת החבילות אינו התקבל אצל הלקוח כתוצאה מנפילות, אומנם השרת קיבל את כלל ההודעות והחל לשלוח *FIN* ועל כן הלקוח קיבל את שתי הודעות ה*FIN* וניתק את החיבור במקום להמשיך ולהמתין.

ip.addr == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	50	54089 → 44444 Len=6
32	5.005535521	127.0.0.1	127.0.0.1	UDP	50	54089 → 44444 Len=6
33	5.005799437	127.0.0.1	127.0.0.1	UDP	50	44444 → 55555 Len=6
34	5.006032437	127.0.0.1	127.0.0.1	UDP	50	55555 → 44444 Len=6
35	5.006252842	127.0.0.1	127.0.0.1	UDP	50	44444 → 54089 Len=6
36	5.006441073	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
37	5.006481529	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
38	7.174398258	127.0.0.1	127.0.0.1	UDP	105	44444 → 55555 Len=61
39	7.174589254	127.0.0.1	127.0.0.1	UDP	105	55555 → 44444 Len=61
44	10.012014900	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
45	10.012053052	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
403	15.017234316	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
404	15.017276259	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
517	20.022478280	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
518	20.022516015	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
523	25.028043480	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
524	25.028082014	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
525	30.033087462	127.0.0.1	127.0.0.1	UDP	144	54089 → 44444 Len=100
526	30.033123930	127.0.0.1	127.0.0.1	UDP	105	54089 → 44444 Len=61
527	30.033308618	127.0.0.1	127.0.0.1	UDP	144	44444 → 55555 Len=100
528	30.033424183	127.0.0.1	127.0.0.1	UDP	105	44444 → 55555 Len=61
529	30.033468777	127.0.0.1	127.0.0.1	UDP	144	55555 → 44444 Len=100
530	30.033568902	127.0.0.1	127.0.0.1	UDP	144	44444 → 54089 Len=100
531	30.033646451	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
532	30.033695633	127.0.0.1	127.0.0.1	UDP	47	55555 → 44444 Len=3
533	30.034320214	127.0.0.1	127.0.0.1	UDP	47	44444 → 54089 Len=3
534	30.034418774	127.0.0.1	127.0.0.1	UDP	47	54089 → 44444 Len=3
535	31.226145365	127.0.0.1	127.0.0.1	UDP	47	44444 → 54089 Len=3