מטלה 4 – רשתות תקשורת

מגישים:

אילן שמחון ת.ז 212036396

תומר גוזלן ת.ז 314770058

רקע

במטלה זו אנו ממשמים תכנית המהווה חלופה לפקודת ping

את התכנית אנו מפעילים בעזרת הפקודה python3 ping.py <ip>

מה שמתרחש כעת הוא שאנו שולחים לכתובת ip שהמשתמש הכניס הודעות שמטרתן לוודא את זמינות ומתן התגובה של כתובת הip המבוקשת, ואנו מצפים לקבל בחזרה מהכתובת הודעה שתאשר לנו שהכל תקין

אנו נשלח את ההודעה שלנו שוב ושוב, ונצפה לקבל את ההודעה מהכתובת שוב ושוב עד שהמשתמש יבחר לעצור את התכנית

בחלק ב, אנו נשדרג את מה שעשינו

תחילה נגדיר כי אם לאחר 10 שניות לא קיבלנו תגובה מהip המבוקש, ככל הנראה הוא לא זמין ולא ניתן ליצור איתו קשר.

לאחר מכן נכתוב קובץ watchdog.py המהווה טיימר שמקבל (בעזרת חיבור tcp) עדכון על כל שליחת הודעה לכתובת ip ועדכון נוסף על כל קבלת תגובה, במידה והטיימר סופר עשר שניות מאז קיבל הודעה על שליחת הודעה והוא טרם עודכן על קבלת תגובה, הוא יתריע על כך ויגרום לping לסגור את התכנית

מהלך הקוד – חלק א'

תחילה נייבא את הספריות הדרושות ונגדיר משתנים

```
import sys
import socket
import os
import struct
import time
import statistics as stats

ICMP_ECHO_REQUEST = 8
ICMP_ECHO_REPLY = 0
PACKET_SIZE = 64
```

כעת, נפעיל את פונקציית הmain שתקרא את כתובת הip מהמשתמש ותשלח אותה לפונקציית send_ping

```
def main():
    if len(sys.argv) != 2:
        print('Correct your command :sudo python3 ping.py <ip_address>')
        exit(1)

    host_ip = sys.argv[1]

    send_ping(host_ip)
```

בתוך הפונקצייה, נגדיר משתנים פנימיים ונפתח את הsocket.

בנוסף, נשנה את ההגדרת כך שהקוד לא ייתקע בהמתנה לתגובה מהיעד אלא ימשיך לשלוח הודעות נוספות (בהתאם למה שפקודת ping המקורית עושה)

```
def send_ping(host_ip):
    times = []

    count_cmp = 0
    count = 0
    count_recev = 0

    s_time = time.time()

    sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)

    sock.setblocking(False)
```

כעת, ניכנס ללולאה הראשית שתשלח ללא הפסקה פאקטה ותחכה לקבל את פאקטת האישור בחזרה מהכתובת המבוקשת

בתחילה, נבנה את הפאקטה, נוודא את שלמות הנתונים באמצעות פונקציית checksum

במידה וזו הפעם הראשונה בה אנו שולחים את ההודעה ליעד, נדפיס למסך פרטים בסיסיים כדוגמת כתובת הip וגודל החבילה הנשלחת, בדומה לפקודת ping

כאן מוגדרת שינה של התכנית לשנייה הכל איטרציה בלולאה, מטרתה היא רק לאזן את קצב שליחת ההודעות ולמנוע וחסר הצלחה לעקוב אחרי כל הפאקטות המתקבלות

כעת, נשלח אל כתובת הip את הפאקטה שלנו ונקרא לפונקצייה receiver_ping על מנת לקבל את התגובה מהיעד

```
try:
    while True:
        count_cmp += 1
        packet = struct.pack('!BBHHH', ICMP_ECHO_REQUEST, 0, 0, 0, count_cmp)
        data = b"this is my massage"

        calc_checksum = checksum(packet + data)

        packet = struct.pack('!BBHHH', ICMP_ECHO_REQUEST, 0, calc_checksum, 0, count_cmp)



        to_send = packet + data
        if count == 0:
            print(f"PING {host_ip} ({host_ip}) {len(packet)}({len(data)}) bytes of data.")

        time.sleep(1)

        sock.sendto(to_send, (host_ip, 1))

        answer = receive_ping(sock,  count_cmp , host_ip)
```

פונקציית checksum

```python
def checksum(data):
    sum = 0
    count_to = (len(data) // 2) * 2
    count = 0

    while count < count_to:
        this_val = data[count+1] * 256 + data[count]
        sum += this_val
        sum &= 0xffffffff
        count += 2

    if count_to < len(data):
        sum += data[len(data) - 1]
        sum &= 0xffffffff

    sum = (sum >> 16) + (sum & 0xffff)
    sum += (sum >> 16)
    result = ~sum & 0xffff
    result = socket.htons(result)

    return result
```

כעת אנו בתוך הפונקצייה receiver_ping

תחילה נגדיר משתנים שנצרכים לנו לפונקצייה, לאחר מכן נקרא לפונקציה recvfrom על מנת לקבל את התגובה מהכתובת, ונשמור את הזמנים שלוקח לנו לקבל את התגובה

כעת, נשמור את הנתונים הנדרשים לנו על מנת להדפיס את התגובה ונוודא כי אכן הפאקטה הגיעה מכתובת הip הנדרשת ואכן מכוונת אלינו (על מנת למנוע מצב בו נקבל פטקטות המשלחות למקום אחר כמו פקודת ping רגילה שיכולה לפעול ברקע)

כעת נוודא גם כי סוג הפאקטה הוא ICMP_ECHO_REPLY ורק לאחר שבירנו שהכל תקין, נדפיס את

התגובה למשתמש בפורמט זהה לחלוטין כמו פקודת ping

```python
def receive_ping(sock, count, host_ip):
    OK = False

    count_rece = 0
    start_time = time.time()
    try:
        recv_packet, addr = sock.recvfrom(PACKET_SIZE + 28)
        icmp_header = recv_packet[20:28]
        icmp_type, code, checksum, packet_id, sequence = struct.unpack(
            "bbHHh", icmp_header
        )

        finish_time = (time.time() - start_time) * 1000

        len_of_packet = len(recv_packet[20:])
        ip_addr = addr[0]
        ttl = recv_packet[8]
        if ip_addr == host_ip and packet_id == os.getgid():
            count_rece += 1

            if icmp_type == ICMP_ECHO_REPLY:
                OK = True
                print(f"{len_of_packet} bytes from {ip_addr}: icmp_seq={count} ttl={ttl} time={finish_time:.3f} ms")
```

כעת נחזור לפונקציית send_ping שמעדכנת כעת את הנתונים אחרי עוד סיבוב מוצלח של שליחת וקבלת פאקטות, ושוב חוזרת לתחילת הלולאה בדרך לסיבוב חדש

במידה והמשתמש מחליט להפסיק את התכנית, נדפיס בדומה לפקודת ping סטטיסטיקה בנוגע לתהליך השליחה שהיה

לאחר מכן, נסגור את החיבור והתכנית תסתיים

```
except KeyboardInterrupt:
    f_time = time.time() - s_time

    print(f"\n--- {host_ip} ping statistics ---")
    print(f"{count} packets transmitted, {count_recev} received,"
          f" {int((1 - (count_recev / count)) * 100)}% packet loss, time {int(f_time * 1000)}ms")
    if count_recev != 0:
        Min = min(times)
        avg = stats.mean(times)
        Max = max(times)
        differences = [abs(time - avg) for time in times]
        mdev = stats.mean(differences)
        print(f"rtt min/avg/max/mdev = {Min:.3f}/{avg:.3f}/{Max:.3f}/{mdev:.3f} ms")

    sock.close()
```

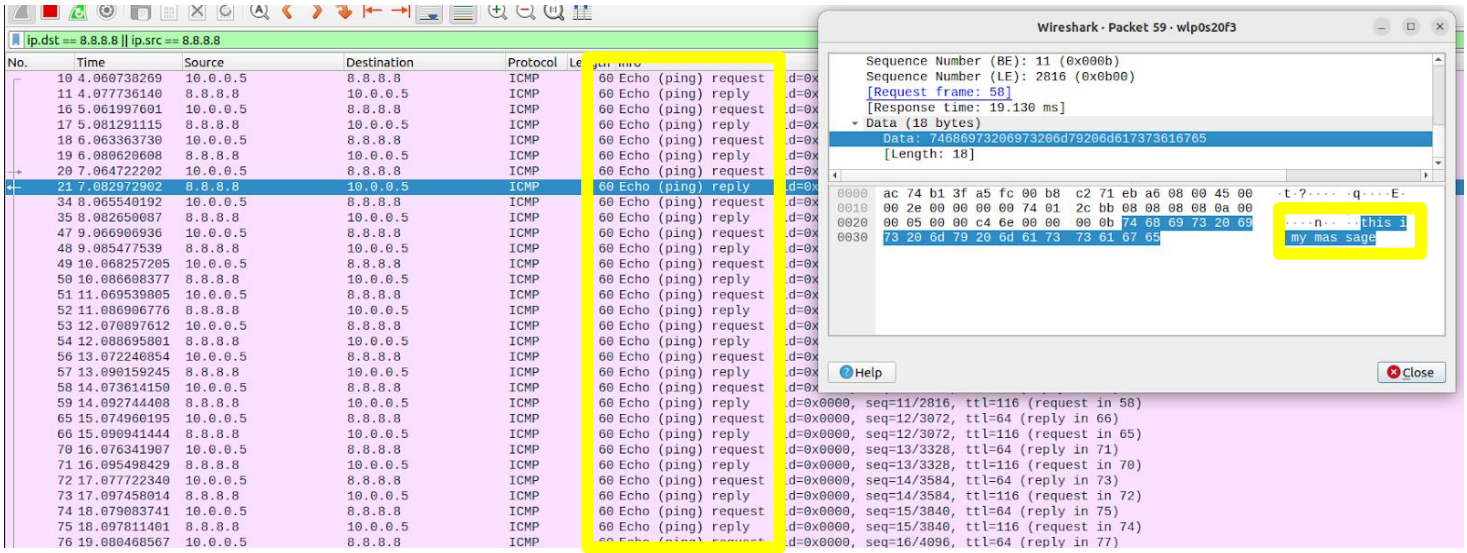
דוגמת הרצה – חלק א

תחילה נריץ על כתובת ip תקינה (למשל של גוגל)

```
ilansimchon@DESKTOP-0K1E6T1:/mnt/c/Users/97253/PycharmProjects/Network4$ sudo python3 ping.py 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 8(18) bytes of data.
26 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=0.029 ms
26 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=0.021 ms
26 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=0.026 ms
26 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=0.029 ms
26 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=0.040 ms
26 bytes from 8.8.8.8: icmp_seq=6 ttl=115 time=0.041 ms
26 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=0.028 ms
26 bytes from 8.8.8.8: icmp_seq=8 ttl=115 time=0.040 ms
26 bytes from 8.8.8.8: icmp_seq=9 ttl=115 time=0.027 ms
26 bytes from 8.8.8.8: icmp_seq=10 ttl=115 time=0.070 ms
26 bytes from 8.8.8.8: icmp_seq=11 ttl=115 time=0.021 ms
^C
--- 8.8.8.8 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 11377ms
rtt min/avg/max/mdev = 0.021/0.034/0.070/0.010 ms
ilansimchon@DESKTOP-0K1E6T1:/mnt/c/Users/97253/PycharmProjects/Network4$
```

ניתן לראות את המידע על כל התגובות שהגיעו מודפסות למסך, ולאחר שהמשתמש סוגר ידנית את התכנית מודפסת סטטיסטיקה של כל החבילות

ניתן לראות את הרצת התכנית גם מwireshark, נבחין כי נשלחת פאקטת request ומייד לאחריה פאקטת reply



כעת נריץ על כתובת ip לא קיימת ונבחין מה קורה

```
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$ sudo python3 ping.py 8.8.8.3
[sudo] password for ilan:
PING 8.8.8.3 (8.8.8.3) 8(18) bytes of data.
^C
--- 8.8.8.3 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss, time 11196ms
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$
```

אכן לא מודפסות הודעות תגובה כלל ולאחר שהמשתמש מפסיק את התכנית רואים כי נשלחו 11 בקשות אך לא התקבלה אף לא תגובה אחת, זאת כיוון שאין אף אחד בכתובת הip של היעד

ניתן לראות זאת גם בwireshark

ip.dst == 8.8.8.3 || ip.src == 8.8.8.3

No.	Time	Source	Destination	Protocol	Length	Info
4	3.416218203	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=1/256, ttl=64 (no response found!)
9	4.417532465	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=2/512, ttl=64 (no response found!)
10	5.418865688	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=3/768, ttl=64 (no response found!)
11	6.420168558	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=4/1024, ttl=64 (no response found!)
20	7.421232489	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=5/1280, ttl=64 (no response found!)
25	8.422523367	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=6/1536, ttl=64 (no response found!)
26	9.423803980	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=7/1792, ttl=64 (no response found!)
27	10.425095601	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=8/2048, ttl=64 (no response found!)
79	11.426339830	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=9/2304, ttl=64 (no response found!)
80	12.427623224	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=10/2560, ttl=64 (no response found!)
84	13.428921159	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=11/2816, ttl=64 (no response found!)

גם כאן ניתן לראות כי בניגוד לצילום הקודם כאן כל הפאקטות הינן של request ואין אף פאקטה של reply

מהלך הקוד – חלק ב'

בחלק זה העתקנו את קובץ הping.py אל קובץ חדש better_ping.py ועשינו בו שינויים על מנת להתאים אותו לעבודה עם הטיימר

בנוסף כתבנו את קובץ הטיימר watchdog.py

נציג כעת את מהלך הקוד בצורה משולבת

תחילה, נגדיר משתנים משותפים לשני הקבצים

```
local_host = '127.0.0.1'
timer_port = 3000

TIMER_ON = "ON"
TIMER_OFF = "OFF"
```

בקובץ הwatchdog ניצור חיבור tcp בינו לבין הbetter_ping

לאחר מכן, נגדיר אותם להימנע מבלוקים על מנת שנוכל לנסות לקבל משהו ואם לא שלחו לי כלום אמשיך הלאה מבלי לחכות, כך אוכל לנסות לקבל שוב ושוב במשך עשר שניות עד שאקבל, ואם לא הצלחתי 10 שניות אתריע על כך

```
def main():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock.bind((local_host, timer_port))

        sock.listen(1)

        timer_sock, timer_addr = sock.accept()

        sock.setblocking(False)
        timer_sock.setblocking(False)
```

במקביל בפונקציית send_ping הbetter_ping מתחבר לחיבור tcp שייצר הwatchdog

```
timer_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

timer_sock.connect((local_host, timer_port))
```

כעת הwatchdig בלולאה יקבל מהbetter_ping ללא הפסקה הודעות על שליחת בקשה שכשהוא יקבל אותם הוא יפעיל את הטיימר, והודעות על קבלת תגובה שיגרמו לו לאפס את הטיימר, בכל איטרציה של הלולאה הוא יבדוק האם הטיימר עבר את 10 השניות ואם כן הוא ישלח על כך התראה לbetter_ping, יסגור את החיבור ויצא מהתכנית

```
while True:
    try:
        status = timer_sock.recv(1024).decode()
    except:
        pass
    if timer_on is False and status == TIMER_ON:
        s_time = time.time()
        timer_on = True
    elif status == TIMER_OFF:
        timer_on = False
    if timer_on and time.time() - s_time > 10:
        timer_sock.send("overtime".encode())
        sock.close()
        timer_sock.close()
        exit(1)


    status = ""
```

כאן ניתן לראות כיצד הbetter_ping שולח הודעה ומייד מעדכן את הטיימר ולאחר שהוא מקבל תגובה ובודק שהיא תקינה הוא מעדכן אותו שוב

```
sock.sendto(to_send, (host_ip, 1))

timer_sock.send(TIMER_ON.encode())

answer = receive_ping(sock,  count_cmp , host_ip)

if answer[1]:
    timer_sock.send(TIMER_OFF.encode())
```

דוגמת הרצה – חלק ב'

תחילה נריץ על כתובת ip תקינה (למשל של גוגל)

```
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$ python3 watchdog.py

(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$ sudo python3 better_ping.py 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 8(18) bytes of data.
26 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.026 ms
26 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=0.014 ms
26 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=0.014 ms
26 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=0.015 ms
26 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=0.017 ms
26 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=0.015 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 7495ms
rtt min/avg/max/mdev = 0.014/0.017/0.026/0.003 ms
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$
```

נראה כי הכל קורה כרגיל, הבקשות והתגובות נשלחות בצורה תקינה ולכן הטיימר אינו משפיע כלל והתכנית רצה עד שהמשתמש יחליט לעצור אותה, נראה זאת גם בwireshark

ip.dst == 8.8.8.8 || ip.src == 8.8.8.8

No.		Source	Destination	Protocol	Le		
10	41.151296393	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=1/256, ttl=64 (reply in 109)
10	41.169033082	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=1/256, ttl=116 (request in 108)
11	42.152519115	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=2/512, ttl=64 (reply in 112)
11	42.170717638	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=2/512, ttl=116 (request in 111)
11	43.153813903	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=3/768, ttl=64 (reply in 114)
11	43.172130414	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=3/768, ttl=116 (request in 113)
11	44.155091408	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=4/1024, ttl=64 (reply in 116)
11	44.173458903	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=4/1024, ttl=116 (request in 115)
11	45.156350697	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=5/1280, ttl=64 (reply in 118)
11	45.173887546	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=5/1280, ttl=116 (request in 117)
11	46.157608427	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=6/1536, ttl=64 (reply in 120)
12	46.174559991	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=6/1536, ttl=116 (request in 119)
12	47.158889305	10.0.0.5	8.8.8.8	ICMP	60	Echo (ping) request	id=0x0000, seq=7/1792, ttl=64 (reply in 126)
12	47.176646850	8.8.8.8	10.0.0.5	ICMP	60	Echo (ping) reply	id=0x0000, seq=7/1792, ttl=116 (request in 125)

ניתן לראות כי הזמנים נמוכים מ10 שניות בין שליחת כל בקשה לתגובה ולכן הטיימר לא משפיע בשום צורה

נריץ כעת על כתובת ip לא פעילה ונבחין מה קורה:

```
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$ python3 watchdog.py
overtime
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$ sudo python3 better_ping.py 8.8.8.3
PING 8.8.8.3 (8.8.8.3) 8(18) bytes of data.
server 8.8.8.3 cannot be reached
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject/Networks-4$
```

אכן התכנית רצה ושלחה בקשות אולם לא קיבלה תגובה ולאחר 10 שניות הטיימר שלח לה התראה לעצור את התכנית ואכן התכנית הופסקה

נראה זאת בwireshark

ip.dst == 8.8.8.3 || ip.src == 8.8.8.3

No.	Time	Source	Destination	Protocol	Length	Info
1	1.847375673	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=1/256, ttl=64 (no response found!)
2	2.848574993	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=2/512, ttl=64 (no response found!)
3	3.849777896	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=3/768, ttl=64 (no response found!)
4	4.850353149	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=4/1024, ttl=64 (no response found!)
5	5.851553286	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=5/1280, ttl=64 (no response found!)
6	6.852764613	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=6/1536, ttl=64 (no response found!)
7	7.853983650	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=7/1792, ttl=64 (no response found!)
8	8.855189045	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=8/2048, ttl=64 (no response found!)
9	9.856403970	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=9/2304, ttl=64 (no response found!)
10	10.857611398	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=10/2560, ttl=64 (no response found!)
11	11.858835452	10.0.0.5	8.8.8.3	ICMP	60	Echo (ping) request id=0x0000, seq=11/2816, ttl=64 (no response found!)

אכן רואים כי מיד כאשר עברו 10 שניות משליחת הבקשה הראשונה הפסיקו להישלח בקשות נוספות וזאת כיוון שהטיימר גרם לעצירת התכנית כמו שהסברנו

דוגמת הרצה נוספת על מנת להראות את פעולת הטיימר

הגדרנו שהתכנית תישן שנייה אחת אחרי שהיא מקבלת את התשובה ולפני שהיא שולחת הודעת אישור לטיימר ובכל סיבוב של שליחת הודעה נוספת הגדלנו את זמן השינה ב3 שניות, וכך ראינו שאכן בהתחלה התכנית ישנה שנייה, ואז 4 שניות, ואז 7 שניות, ובכל הזמן הזה הכל המשיך לעבוד כראוי.

ואז התכנית ישנה 10 שניות ושם הטיימר קפץ והתכנית הופסקה עם הודעה שאי אפשר ליצור קשר עם כתובת זו (כמובן שזה לא נכון אלא רק בגלל מה שהגדרנו)

```
ilansimchon@DESKTOP-0K1E6T1:/mnt/c/Users/97253/PycharmProjects/Network4$ sudo python3 better_ping.py 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 8(18) bytes of data.
26 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=0.036 ms
26 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=0.026 ms
26 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=0.028 ms
26 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=0.029 ms
server 8.8.8.8 cannot be reached
```

(כמובן שאין בתכנית זו איזה היגיון או שימוש אלא היא נועדה רק להמחיש את פעולת הטיימר וכמובן שלאחר ההרצה שלה מחקנו את השינה מהקוד)

סיכום

הצלחנו לממש את הפקודה ping בעצמנו ולשלוח בקשה לכל ip שנרצה, כמו כן עקבנו אחרי התהליך המקורי של הפקודה ping וחיקינו אותו במדויק הן בהדפסות, הן בסטטיסטיקה שאחרי סיום התכנית, והן במניעת חרגיות (כגון הפעלת הפקודה ping במקביל לקובץ ping.py, זה עלול להוביל לקבלת תגובות שלא מיועדות אלינו ובאגים נוספים)

בנוסף, הצלחנו לשפר את קובץ ה ip שלנו כך שיזהה כתובות ip שאינם קיימות ויפסיק את הניסיון לשלוח להם בקשות

ראינו את שליחת הפאקטות בwireshar והבחנו בפרטים הקטנים הנדרשים כמו עצירה אחרי 10 שניות וכו'