

# דו"ח מסכם לפרויקט גמר: ניתוח תבעורה בפרוטוקול IP/TCP

קורס רשות תקשורת מחשבים (0061305-03)

**מגייסים:** אלמוג שורצברג

אליעד ביצ'ר

**מרצה:** ד"ר אנדרי קוזוכוב

סמסטר א', תשפ"ז  
ינואר 2026

## תוכן עניינים

2 .....	חלק 1: ניתוח תעבורות פאקטוות בפרוטוקול TCP/IP
2 .....	שלב 1 – הכנות קובץ CSV
3 .....	שלב 2 – אריזת הנתונים (Encapsulation)
3 .....	שלב 3 – לכידה ב-Wireshark
5 .....	חלק 2: פיתוח יישום צ'אט
5 .....	הסבר כללי על המערכת:
6 .....	הוראות התקינה והרצתה:
7 .....	דוגמה להרצת היישום:
9 .....	מבנה הקוד - קובץ client.py:
10 .....	מבנה הקוד - קובץ server.py:
11 .....	ניתוח תעבורת רשות ביישום

## חלק 1: ניתוח תעבורת פקודות ב프וטוקול TCP/IP

ב חלק זה נמיר הودעות בשכבה היחסום לפקודות IP/TCP ונכוד את התעבורה שלhn ב-Wireshark. המטרה היא להבין האם הפקודות בכלל נקלטות, משודרות בצורה תקינה ונקלטות במלואן. בהמשך, נבון איך הן נראהות.

לבסוף נסביר את הממצאים בהתבסס על קובץ CSV המקורי שמכיל את ההודעות.

### שלב 1 – הכנת קובץ CSV

בשלב זה הכנו קובץ CSV עם הודעות משכבה האפליקציה שמדוות תעבורת בפרוטוקול HTTP בעזרת בינה מלאכותית **Gemini** (מצורף בתמונה למטה).

המאפיינים של כל הודעה הם (משמאל לימין): מספר סידורי, סוג ה프וטוקול, מקור, יעד, פורט של המקור, פורט של היעד, תוכן הודעה, ייחידת הזמן.

msg_id	app_protocol	src_app	dst_app	src_port	dst_port	message	timestamp
1	HTTP	client_browser	web_server	51874	80	GET /index.html HTTP/1.1	0.010
2	HTTP	web_server	client_browser	80	51874	HTTP/1.1 200 OK	0.025
3	HTTP	client_browser	web_server	51874	80	GET /style.css HTTP/1.1	0.040
4	HTTP	client_browser	web_server	51874	80	GET /script.js HTTP/1.1	0.055
5	HTTP	web_server	client_browser	80	51874	HTTP/1.1 200 OK (CSS file)	0.070
6	HTTP	web_server	client_browser	80	51874	HTTP/1.1 200 OK (JS file)	0.085
7	HTTP	client_browser	web_server	51874	80	POST /login HTTP/1.1	0.120
8	HTTP	web_server	client_browser	80	51874	HTTP/1.1 302 Found (Redirect)	0.150
9	HTTP	client_browser	web_server	51874	80	GET /dashboard HTTP/1.1	0.180
10	HTTP	web_server	client_browser	80	51874	HTTP/1.1 200 OK (Dashboard)	0.210

## שלב 2 – אריזת הנטוונים (Encapsulation)

השתמשנו במחברת ה-Jupyter על מנת לארח את ההודעות לפקודות TCP/IP.

1. הקוד יוצר IP Header שכולל: כתובות IP של המקור והיעד, identification, ttl, checksum ועוד.

2. הקוד גם יוצר TCP Header שכולל: פורטים מקור, פורט יעד, sequence number ועוד.

3. לבסוף הקוד לחבר את הכל כולל ה-payload לכדי פקחת TCP/IP שלמה.

בתמונה למטה ניתן לראות את מבנה הפקטה בפורמט הקסדצימלי.

```
In [22]: # Preview packet structure
src_ip = '127.0.0.1'
dst_ip = '127.0.0.1'
src_port = random.randint(1024, 65535)
dst_port = 12345
payload = b'Hello Packet (preview)'
pkt_preview = build_ip_header(src_ip, dst_ip, 20 + len(payload)) + build_tcp_header(src_ip, dst_ip, src_port, dst_port, payload)
hexdump(pkt_preview)
```

0000 45 00 00 3e 3b 03 00 00 40 06 41 b5 7f 00 00 01 E..>...@A....  
0010 7f 00 00 01 f8 64 30 39 e6 64 e0 c2 00 00 00 00 .....d09.d.....  
0020 50 02 ff ff ae 31 00 00 48 65 6c 6c 6f 20 50 61 P....1..Hello Pa  
0030 63 6b 65 74 20 28 70 72 65 76 69 65 77 29 cket (preview)

## שלב 3 – לכידה ב-Wireshark

כעת כישש לנו את הפקטות מוכנות, שלחנו אותן כדי לראות אותן ב-Wireshark ולנתה:

1. באמצעות Wireshark האזנו למשק Loopback (היעד והמקור הם אותו המחשב וכך).  
השימוש במשק ה-Loopback.

2. הרצינו במחברת ה-Jupyter את הקוד ששולח את הפקטות. הקוד עובר על קובץ CSV  
שהכנו ושולח את ההודעות אחת אחרי השניה.

בתמונה למטה ניתן לראות את הפקטות שלנו ב-Wireshark. נפרט את התוצאות.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	68	45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=24
2	0.000058	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
3	0.181530	127.0.0.1	127.0.0.1	TCP	59	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
4	0.181566	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
5	0.202938	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
6	0.202981	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
7	0.304395	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
8	0.304428	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
9	0.406783	127.0.0.1	127.0.0.1	TCP	70	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=26
10	0.406812	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
11	0.508749	127.0.0.1	127.0.0.1	TCP	69	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=25
12	0.508780	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
13	0.610196	127.0.0.1	127.0.0.1	TCP	64	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=20
14	0.610227	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
15	0.711918	127.0.0.1	127.0.0.1	TCP	73	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=29
16	0.711947	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
17	0.813508	127.0.0.1	127.0.0.1	TCP	67	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=23
18	0.813543	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0
19	0.915309	127.0.0.1	127.0.0.1	TCP	71	[TCP Retransmission] 45234 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=27
20	0.915335	127.0.0.1	127.0.0.1	TCP	44	12345 → 45234 [RST] Seq=1 Win=0 Len=0

שורה 1: ה"לקוח" מנסה לשלוח מידע באורך Len=24 ל"שרת" בפורט 12345, לחבילה מצורף עם דגל [PSH, ACK] (במקרה זה התוכן הוא GET /index.html HTTP/1.1).

שורה 2: המחשב מחזיר ל"לקוח" חבילה עם דגל RST (כפי שכתו בעמודה INFO בחילון הWIRESHARK), ככלומר הוא מצין בפנוי הלקוח כי הפורט 12345 סגור ואין אף process במחשב שמאזין לפורט זהה ומורה לו להפסיק את הניסיונות לשЛОוח את החבילה.

שורה 3: ה"לקוח" מנסה לשלוח מידע באורך Len=15 ל"שרת" בפורט 12345, לחבילה מצורף עם דגל [ACK] (במקרה זה התוכן הוא HTTP/1.1 200 OK). **WIRESHARK מזהה בחבילה את TCP Retransmission Sequence Number שנשלח בעבר ולכן מניחה כי מדובר בטעות** של **אותה הודעה** - למרות העובדה שמדובר בהודעה חדשה שהלkoוח שלח לפורט זהה.

שורה 4: המחשב מחזיר שוב ל"לקוח" חבילה עם דגל RST כי הפורט 12345 סגור ואין אף process במחשב שמאזין לפורט זהה.

lolaha zo chzart ul uzma b'shorot 5-20 b'WIRESHARK. lkooch matkdm b'hodutot v'misorev pum achor pum u'i mchshab bgel shahfort 12345 sgor. bgel shmchbarat jupyter la manhalat at kidom h sequence number b'khol hoduta chadsha shnashlach, WIRESHARK mnihah ci uzem ha'oubda shehmaspfer nshar 1 mahooh shidur chzr shel hodutot (mahe shbafouel la nkon).

## **חלק 2: פיתוח יישום צ'אט**

### **הסבר כללי על המערכת:**

הישום מבוסס על ארכיטקטורת שרת-לקוח שבו השרת הוא רכיב עם כתובת IP קבועה ש"תמיד פועל", ככלומר שמחכה ומאזין לפקודות של לקוחות. הלקוחות הם שיחמים את החיבור מולו. התקשרות ביניהם מתבצעת עם TCP Sockets כממשק בין שכבה האפליקציה לשכבה התעבורה.

הסיבה לשימוש ב프וטוקול TCP בישום זה היא הצורך בהעברת נתונים אמינים. ככלומר אנחנו נדרשים להבטיח שכל ההודעות של הלקוחות הגיעו ליעד, בסדר הנכון ומבלתי לאבד מידי.

הישום מחולק לשני קבצי קוד בשפת פיתון: `server.py` ו-`client.py`.

## הוראות התקינה והרצתה:

1. וודאו שביבת פיתון מותקנת על המחשבים בהם ירצו השירות והלקוחות.
2. התקינו את הקבצים `client.py` ו- `server.py`. לאחר התקינה עליהם להימצא באותה תיקייה.
3. **לא חובה** - כדי להריץ את הלקוחות והשרות על **מחשבים שונים באותה רשת** עליהם לשנות את ערכי הקבועים `SERVER_IP` ו- `SERVER_PORT` בקובץ `server.py` לכתובת IP והפרטיה של המחשב שיריץ את השירות ולפורט פנוי כלשהו. בקובץ `client.py` יש להתאים בדיק את אותם ערכים.

```
##### Set SERVER IP to the private IP address of the computer that
will run the server #####
##### Set PORT to any available port that isn't in use by another ser-
vice #####
SERVER_IP = "localhost"
SERVER_PORT = 12000
```

4. לחזו מקש ימני על הקובץ `server.py` ובחרו באופציית `copy as path`.
5. פתחו חלון CMD במחשב שיריץ את השירות. בחلون זה יש להדביק את הטקסט בלחיצת על `Alt+ENTER` ומיד ללחוץ `ENTER`. הקוד של השירות יירוץ והשרות ידפיס הודעה המאשרת שהוא במצב ONLINE וממתין לחיבור ללקוחות.
6. עברו לקו חדש: פתחו חלון CMD חדש במחשב שיריץ את הלקוחות.
7. לחזו מקש ימני על הקובץ `client.py` ובחירה באופציית `copy as path`. יש להדביק בCMD את הטקסט בלחיצת על `Alt+ENTER` ומיד ללחוץ `ENTER`.
8. הקוד של הלקוח יירוץ ויבוצע חיבור לשרת. לאחר הצלחת החיבור, הלקוח יתבקש להזין שם משתמש. יש להזין שם משתמש ייחודי לכל לקוח בעת החיבור. **חזרו על צעדים 7-6 אם ברצונכם להריץ מספר לקוחות.**
9. לשילוח הודעות בין לקוחות יש להשתמש בפורמט הטקסטואלי `<@target <message>>`.  
לדוגמה, אם ברצנו לשלוח Hey dude Bob למשתמש Bob נרשם `@Bob Hey dude`

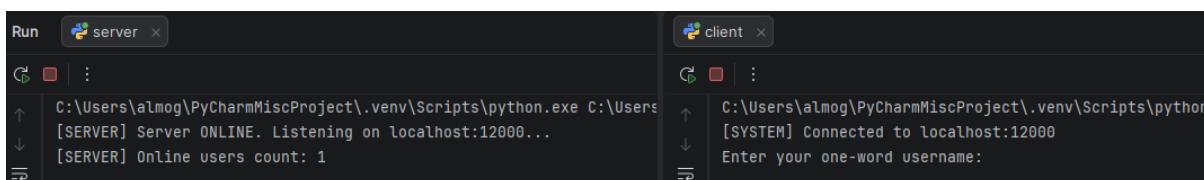
## דוגמיה להרצת היישום:

נרים את server.py ונקבל את הודעה הבאה שימושה שהשרת פועל ומאזין לחיבורים:



```
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\server.py
[SERVER] Server ONLINE. Listening on localhost:12000...
```

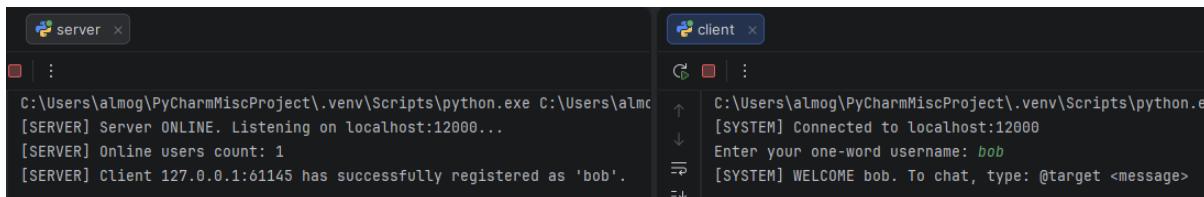
מיד לאחר מכן נרים דרך טרמינל את client.py . נשים לב כי מס' המשתמשים המוחברים התעדכן בשרת ל-1, ובחלון הclient נקבל הודעה הצלחה על חיבור לשרת ונתבקש להזין שם משתמש:



Run	server
ɔ	:
↑	C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\server.py
↓	[SERVER] Server ONLINE. Listening on localhost:12000...
≡	[SERVER] Online users count: 1

ɔ	client
ɔ	:
↑	C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\client.py
↓	[SYSTEM] Connected to localhost:12000
≡	Enter your one-word username:

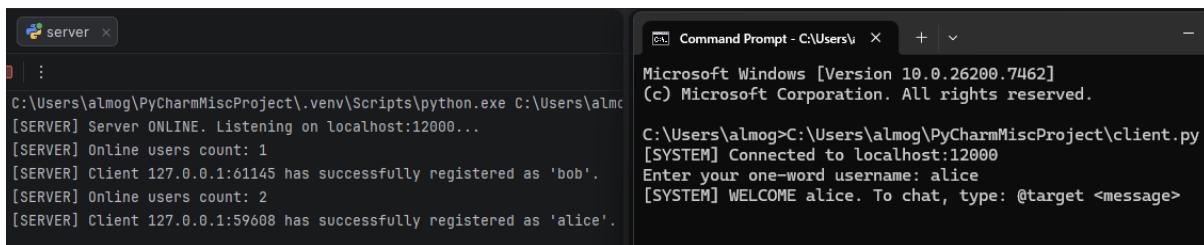
ນבחר בשם bob, ונקבל פלט מתאים בחלון client וערכו בשרת כי נוצר משתמש חדש:



ɔ	server
ɔ	:
↑	C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\server.py
↓	[SERVER] Server ONLINE. Listening on localhost:12000...
≡	[SERVER] Online users count: 1
≡	[SERVER] Client 127.0.0.1:61145 has successfully registered as 'bob'.

ɔ	client
ɔ	:
↑	C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\client.py
↓	[SYSTEM] Connected to localhost:12000
≡	Enter your one-word username: bob
≡	[SYSTEM] WELCOME bob. To chat, type: @target <message>

בצורה דומה נתחבר מטרמינל חדש עם משתמש ששמו alice ונקבל פלט מתאים ללקוח זה, וכן:  
עדכו בשרת:



ɔ	server
ɔ	:
↑	C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\server.py
↓	[SERVER] Server ONLINE. Listening on localhost:12000...
≡	[SERVER] Online users count: 1
≡	[SERVER] Client 127.0.0.1:61145 has successfully registered as 'bob'.
≡	[SERVER] Online users count: 2
≡	[SERVER] Client 127.0.0.1:59608 has successfully registered as 'alice'.

ɔ	Command Prompt - C:\Users\almog
ɔ	Microsoft Windows [Version 10.0.26200.7462]
ɔ	(c) Microsoft Corporation. All rights reserved.
ɔ	C:\Users\almog>C:\Users\almog\PyCharmMiscProject\client.py
ɔ	[SYSTEM] Connected to localhost:12000
ɔ	Enter your one-word username: alice
ɔ	[SYSTEM] WELCOME alice. To chat, type: @target <message>

כעת bob ישלח הודעה אל alice שתוכנה: Hey, how are you doing today? באמצעות הפורט הטקסטואלי. הודעה תירשם כוגה גם בשרת:

```

server : 
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscP
[SERVER] Server ONLINE. Listening on localhost:12000...
[SERVER] Online users count: 1
[SERVER] Client 127.0.0.1:61145 has successfully registered as 'bob'.
[SERVER] Online users count: 2
[SERVER] Client 127.0.0.1:59608 has successfully registered as 'alice'.
[LOG] bob (127.0.0.1:61145) --> alice (127.0.0.1:59608): Hey, how are you doing today?

client : 
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscP
[SYSTEM] Connected to localhost:12000
Enter your one-word username: bob
[SYSTEM] WELCOME bob. To chat, type: @  
@alice Hey, how are you doing today?

Command Prompt - C:\Users\almog>
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.
C:\Users\almog>C:\Users\almog\PyCharmMiscProject\client>
[SYSTEM] Connected to localhost:12000
Enter your one-word username: alice
[SYSTEM] WELCOME alice. To chat, type: @target <message>
Message from bob: Hey, how are you doing today?

```

כעת alice תתנתק מהיישום מרצוניה החופשי ע"י הזנת הפקודה exit, הפעולה תירשם כוגה בשרת:

```

server : 
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscP
[SERVER] Server ONLINE. Listening on localhost:12000...
[SERVER] Online users count: 1
[SERVER] Client 127.0.0.1:61145 has successfully registered as 'bob'.
[SERVER] Online users count: 2
[SERVER] Client 127.0.0.1:59608 has successfully registered as 'alice'.
[LOG] bob (127.0.0.1:61145) --> alice (127.0.0.1:59608): Hey, how are you doing today?
[SERVER] Client 127.0.0.1:59608 (username: 'alice') has disconnected from server.

client : 
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscP
[SYSTEM] Connected to localhost:12000
Enter your one-word username: alice
[SYSTEM] WELCOME alice. To chat, type: @  
@alice Hey, how are you doing today?

Command Prompt - C:\Users\almog>
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.
C:\Users\almog>exit
[SYSTEM] Disconnected from server.

```

דוגמה נוספת להרצת היישום עם 5 לקוחות, מזויתו של השרת:

```

Run server :
server : 
C:\Users\almog\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\almog\PyCharmMiscProject\server.py
[SERVER] Server ONLINE. Listening on localhost:12000...
[SERVER] Online users count: 1
[SERVER] Client 127.0.0.1:55862 has successfully registered as 'eliad'.
[SERVER] Online users count: 2
[SERVER] Client 127.0.0.1:51738 has successfully registered as 'tal'.
[SERVER] Online users count: 3
[SERVER] Client 127.0.0.1:63357 has successfully registered as 'david'.
[SERVER] Online users count: 4
[SERVER] Client 127.0.0.1:63399 has successfully registered as 'walter'.
[SERVER] Online users count: 5
[SERVER] Client 127.0.0.1:57780 has successfully registered as 'jesse'.
[LOG] david (127.0.0.1:63357) --> jesse (127.0.0.1:57780): hi man
[LOG] tal (127.0.0.1:51738) --> walter (127.0.0.1:63399): lovely day, isn't it?
[LOG] jesse (127.0.0.1:57780) --> david (127.0.0.1:63357): hi, just saw your message, how are ya?
[LOG] eliad (127.0.0.1:55862) --> tal (127.0.0.1:51738): can i borrow $9999999999 ?
[SERVER] Client 127.0.0.1:55862 (username: 'eliad') has disconnected from server.
[SERVER] Client 127.0.0.1:63357 (username: 'david') has disconnected from server.
[SERVER] Client 127.0.0.1:63399 (username: 'walter') has disconnected from server.
[SERVER] Client 127.0.0.1:51738 (username: 'tal') has disconnected from server.
[SERVER] Client 127.0.0.1:57780 (username: 'jesse') has disconnected from server.

```

## מבנה הקוד - קובץ client.py

מהו זה妾 הלקוח של היישום. הלקוח יוזם חיבור לכתובת השירות (בעזרת קבועים IP\_SERVER ו- PORT שגדירים לפני הרצה).

כשהוא מתחבר, הלקוח נכנס ללולאה שבה הוא מتابקש להזין שם משתמש כדי להשתמש ביישום. הלולאה תבצע עד לקבלת אישור מהשירות (response == "WELCOME").

```
while True:
    name = input("Enter your one-word username: ").strip()
    client.sendall(name.encode("utf-8"))

    response = client.recv(1024).decode("utf-8")

    if response == "INVALID_NAME":
        print("[ERROR] Username is invalid. Make sure it is a single word and not empty.")
    elif response == "NAME_TAKEN":
        print("[ERROR] Username is already taken. Retry with another name.")
    elif response == "WELCOME":
        print(f"[SYSTEM] WELCOME {name}. To chat, type: @target <message>")
        break
```

לאחר מכן הלקוח נוצר thread האזנה (fonkciyah listen\_for\_messages) במקביל לthread המרכז שבו הלקוח מتابקש להזין הודעות כדי לצוות עם לקוחות אחרים שמחוברים ליישום.

**fonkciyah listen for messages:** הלקוח נכנס ללולאת האזנה לשרת. כשמקבלת תגובה מהשירות, מודפס פלט למסך של הלקוח בהתאם לקלט (ההודעות) של אותו לקוח.

```
while True:
    try:
        msg = client_socket.recv(1024).decode("utf-8")
        if not msg:
            break

        if msg == "CANT_MESSAGE_SELF":
            print("[ERROR] You cannot send messages to yourself!")
        elif msg == "WRONG_USAGE":
            print(f"[SYSTEM] Usage: @name <message>")
        else:
            print(f"\r{msg}\n", end="")
            # prints the incoming messages to the client
    except (ConnectionResetError, KeyboardInterrupt, ConnectionAbortedError):
        break
```

לאורך כל התכנית,妾 הלקוח יודע להתמודד עם ניתוקים לא צפויים ועדכן את הלקוח ואת השירות עלacr בזמן אמת.

## מבנה הקוד - קובץ :server.py

מהו זה בצד השרת של היישום. הוא יוצר Socket מסוג SOCK\_STREAM (כלומר שימוש בפרוטוקול TCP), מבצע bind לפורט ספציפי ומאזין לבקשת כניסה. הצד השרת בונה על מודל ה-threading: עבור כל לקוח חדש שמחבר, השרת יוצר thread נפרד שמקבל את Socket של הלקוח ושלח להריץ את פונקציית `handle_client_input`.

**פונקציית handle\_client\_input**: השרת נכנס ללולאה עד לקבלת שם משתמש תקין מהלקוח, רק לאחר הצלחה הלקוח נשמר במלון, נשלחת הודעה WELCOME והלולאה מסתירה.

מיד לאחר מכן השרת נכנס שוב ללולאת האזנה ללקוח ספציפי. השרת ממתין להודעות שהלקוח מעוניין לשלוח למשתמשים אחרים שמחוברים גם הם. כשמתקבלת הודעה מהמשתמש, השרת מחפש את שמו של הנמען (`target_name`) במלון ושולח בsocket שלו את הודעה (כלומר בפועל השרת הוא המתווך בין הלקוחות - לקוח A שירצה לשלוח הודעה ללקוח B, למעשה ישלח את הודעה לשרת, והשרת הוא זה שישלח את הודעה ללקוח B).

```
while True:
    data = client_socket.recv(1024).decode('utf-8')
    if not data or data == "EXIT":
        break

    if data.startswith("@") and " " in data:
        # 1st character of data is @, so skip it with data[1:]
        # cut by 1st space so the 1st word = name, after 1st word - msg
        parts = data[1:].split(" ", maxsplit=1)
        target_name = parts[0]
        message_content = parts[1]

        if target_name == username:
            client_socket.sendall(f"CANT_MESSAGE_SELF".encode("utf-8"))
            continue

        with clients_lock:

            if not target_name in online_clients:
                client_socket.sendall(f"[SYSTEM]: User '{target_name}' was
not found.".encode("utf-8"))
                continue

            target_socket = online_clients[target_name]["socket"]
            target_ip = online_clients[target_name]["address"]

            target_socket.sendall(f"Message from {username}: {message_content}"
".encode("utf-8"))
                print(f"[LOG] {username} ({formatted_addr}) -> {target_name}
({target_ip}): {message_content}")
            else:
                client_socket.sendall("WRONG USAGE".encode("utf-8"))
```

לאור כל התכנית, הצד השרת ידע להתמודד עם ניתוקים לא צפויים ועדכן על כך בזמן אמת.

## ניתוח תעבורת רשת ביחס

לצורך בוחנת התעבורה בפרויקט, הפעלנו שרת מחשב אחד ו 2 לköחות מחשב אחר. שני המחשבים פועלים באותה הרשת. את לכידת הפקאות ביצענו Wireshark ושמרנו לתוך קובץ server-client-messages.pcap המצורף לפרויקט.

נסביר את התעבורה לפי שורות שמוספרות בעמודה No בתמונה למטה:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	66	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52
6	2.262556	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=6 Ack=53 Win=65280 Len=0
7	234.572143	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
8	234.572235	192.168.0.186	192.168.0.165	TCP	66	10000 → 55877 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
9	234.573794	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
10	238.101365	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=6
11	238.101598	192.168.0.186	192.168.0.165	TCP	107	10000 → 55877 [PSH, ACK] Seq=1 Ack=7 Win=2097920 Len=53
12	238.152258	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=7 Ack=54 Win=65280 Len=0
13	291.451842	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [PSH, ACK] Seq=7 Ack=54 Win=65280 Len=12
14	291.451942	192.168.0.186	192.168.0.165	TCP	80	10000 → 55841 [PSH, ACK] Seq=53 Ack=6 Win=2097920 Len=26
15	291.498389	192.168.0.186	192.168.0.165	TCP	54	10000 → 55877 [ACK] Seq=54 Ack=19 Win=2097920 Len=0
16	291.500110	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=6 Ack=79 Win=65280 Len=0

שורה 1: מתחילה TCP Handshake. לקוח ראשון יוזם חיבור לשרת (שכבר פועל וממתין לחברורים). הלקוח שלוח פקאה עם דגל SYN מופעל. משורה זו מבנים כי IP של הלקוח והפורט שלו הם לקוח שלח פקאה עם דגל SYN (זהו ה IP של השרת הוא 192.168.0.186 (SOURCE) וכי ה IP של השרת הוא 192.168.0.165:55841 (DESTINATION).

שורה 2: השלב השני של TCP Handshake. השרת עונה "קיבלתني את הבקשה שלך (ACK), ואני מוכן להתחבר (SYN)".

שורה 3: הלקוח מאשר (ACK). החיבור נוצר רשמי. זהו השלב האחרון ב TCP Handshake.

שורה 4: הלקוח שלוח את שם המשתמש שלו לשרת (Len=5). במקרה זה, בחרנו בשם eliad.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	66	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52

```

> Frame 4: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: Intel_e2:98:b1 (18:93:41:e2:98:b1), Dst: GigaByteTech_2f:7d (08:00:2f:7d:00:00)
> Internet Protocol Version 4, Src: 192.168.0.165, Dst: 192.168.0.186
> Transmission Control Protocol. Src Port: 55841. Dst Port: 10000. Seq: 1. Ack: 1
0000 18 c0 4d 2f 74 5c 18 93 41 e2 98 b1 08 00 45 00  M/t\... A...E...
0010 00 2d 8f fc 40 00 80 06 e8 1e c0 a8 00 a5 c8 a8  ...
0020 00 ba da 21 27 10 6d e6 57 a0 23 21 2f fb 50 18  ...
0030 00 ff df 75 00 00 65 6c 69 61 64 8e  u eliad

```

**שורות 5:** השרת שולח הודעה WELCOME שאורכה 52

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	66	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52

שורה 6: הלקוח מאשר (ACK) שקיבל את הודעה WELCOME של השרת.

**שורה 7: מאותו מחשב של הלקובח eliad** הרצנו לקווח חדש שיזם חיבור לשרת (הווכחה לכך - SOURCE זהה ל-SOURCE שיזם חיבור לשרת בשורה 1). הפורט של הלקובח החדש הוא 55877. מתחילה תהילר נוסף של TCP Handshake.

שורה 8: השלב השני של TCP Handshake. השרתעונה "קיבלת את הבקשה שלך (ACK), ואני מוכן להתחבר (SYN)".

שורה 9: הלקוח מאשר (ACK). החיבור נוצר רשמי. זהו השלב האחרון ב-TCP Handshake.

שורה 10: הלקוק החדש שולח את שם המשתמש שלו לשרת שאורךו 6=Len. במקרה זה, בחרנו בשם laptop.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52
6	2.262556	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=6 Ack=53 Win=65280 Len=0
7	234.572143	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
8	234.572235	192.168.0.186	192.168.0.165	TCP	66	10000 → 55877 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
9	234.573794	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
10	238.101365	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=6

---

> Frame 10: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits)  
 > Ethernet II, Src: Intel\_e2:98:b1 (18:93:41:e2:98:b1), Dst: GigaByteTech\_2f:74  
 > Internet Protocol Version 4, Src: 192.168.0.165, Dst: 192.168.0.186  
 > Transmission Control Protocol, Src Port: 55877, Dst Port: 10000, Seq: 1, Ack:

0000	18	c0	4d	2f	74	5c	18	93	41	e2	98	b1	08	00	45	00	..	M/t	A	E
0010	00	2e	90	00	40	00	80	06	e8	19	c0	a8	00	a5	c0	a8	..	@	.	.
0020	00	ba	45	27	10	c6	c1	24	dd	03	5a	21	c0	50	18	..	E	.	\$	Z
0030	00	ff	c2	c2	00	86	6c	61	70	74	f6	78	..	..	..	..	la	ptp	.	.

שורה 11: השרת שולח הודעה WELCOME ל- laptop שאורכה Len=53

**שורה 12:** הלקוק laptop מאשר (ACK) שקיבל את הודעה WELCOME של השירות.

שורה 13: הליקות laptop רוצה לשЛОח הودעה באורך Len=12 ללקוח eliad שהתוכן שלו הוא hello. הוא מעביר את ההודעה לשרת שמשמש כמתווך. זיהינו שמדובר בלקוח laptop מכיוון שהפורט של המחשב השולח את הפקטה הוא 55877 מהתבוננות בעמודה הימנית info שבתמונה:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	66	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52
6	2.262556	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=6 Ack=53 Win=65280 Len=0
7	2.34.572143	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
8	2.34.572235	192.168.0.186	192.168.0.165	TCP	66	10000 → 55877 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
9	2.34.573794	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
10	2.38.101365	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=6
11	2.38.101598	192.168.0.186	192.168.0.165	TCP	107	10000 → 55877 [PSH, ACK] Seq=1 Ack=7 Win=2097920 Len=53
12	2.38.152258	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=7 Ack=54 Win=65280 Len=0
13	291.451842	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [PSH, ACK] Seq=7 Ack=54 Win=65280 Len=12

```

> Frame 13: Packet, 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Intel_e2:98:b1 (18:93:41:e2:98:b1), Dst: GigaByteTech_2f:74
> Internet Protocol Version 4, Src: 192.168.0.165, Dst: 192.168.0.186
> Transmission Control Protocol, Src Port: 55877, Dst Port: 10000, Seq: 7, Ack:
  ▾ Data (12 bytes)
    0000  18 c0 4d 2f 74 5c 18 93 41 e2 98 b1 08 00 45 00  ··M/t\.. A...E
    0010  00 34 90 02 40 00 80 06 e8 11 c0 a8 00 a5 c0 a8  ··4...@...
    0020  00 ba da 45 27 10 c6 c1 24 e3 03 5a 21 f5 50 18  ..E... $ Z! P
    0030  00 ff 19 51 00 00 40 65 6c 69 61 64 20 68 65 6c  ..Q..@eliad hel
    0040  6c 6f                                         lo
  ▾ Data (12 bytes)

```

שורה 14: השרת מעביר את ההודעה ללקוח בפורט 55841, כלומר ללקוח eliad . אורך ההודעה גדול ל Len=26 כי בקוד הוספנו את הקידומת Message from user:

No.	Time	Source	Destination	Protocol	Length	Info
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
3	0.001501	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	2.215941	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
5	2.216187	192.168.0.186	192.168.0.165	TCP	106	10000 → 55841 [PSH, ACK] Seq=1 Ack=6 Win=2097920 Len=52
6	2.262556	192.168.0.165	192.168.0.186	TCP	60	55841 → 10000 [ACK] Seq=6 Ack=53 Win=65280 Len=0
7	2.34.572143	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
8	2.34.572235	192.168.0.186	192.168.0.165	TCP	66	10000 → 55877 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_
9	2.34.573794	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
10	2.38.101365	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=6
11	2.38.101598	192.168.0.186	192.168.0.165	TCP	107	10000 → 55877 [PSH, ACK] Seq=1 Ack=7 Win=2097920 Len=53
12	2.38.152258	192.168.0.165	192.168.0.186	TCP	60	55877 → 10000 [ACK] Seq=7 Ack=54 Win=65280 Len=0
13	291.451842	192.168.0.165	192.168.0.186	TCP	66	55877 → 10000 [PSH, ACK] Seq=7 Ack=54 Win=65280 Len=12
14	291.451942	192.168.0.186	192.168.0.165	TCP	80	10000 → 55841 [PSH, ACK] Seq=53 Ack=6 Win=2097920 Len=26

```

> Frame 14: Packet, 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
> Ethernet II, Src: GigaByteTech_2f:74:5c (18:04:4d:2f:74:5c), Dst: Intel_e2:98
> Internet Protocol Version 4, Src: 192.168.0.186, Dst: 192.168.0.165
> Transmission Control Protocol, Src Port: 10000, Dst Port: 55841, Seq: 53, Ack:
  ▾ Data (26 bytes)
    0000  18 93 41 e2 98 b1 18 c0 4d 2f 74 5c 08 00 45 00  ..A.... M/t\.. E-
    0010  00 42 59 27 40 00 80 06 1e df c0 a8 00 ba c0 a8  BY'@...
    0020  00 27 10 da 21 23 21 30 2f 6d e5 57 a5 50 18  ..'..#!@ m-W.P.
    0030  20 03 3c e7 00 00 4d 65 73 73 61 67 65 20 66 72  <.. Me ssage fr
    0040  6f 6d 20 6c 61 70 74 6f 70 3a 20 68 65 6c 6c 6f  om laptop: hello
  ▾ Data (26 bytes)

```

שורה 15: השרת מאשר (ACK) ללקוח laptop (ACK) ללקוח eliad שהים שההודעה ע"פ שורה 14 שבתמונה בעמ' 10 שהוא קיבל את תוכן ההודעה שרצוים להעביר ללקוח eliad (בפועל השרת כבר העביר את ההודעה ע"פ שורה 14 ושלח אישור באיחור).

שורה 16: הליקוח eliad מאשר (ACK) שקיבל את ההודעה שהשרת העביר לו (זהים שההודעה eliad לפוּרְט 55841 שבתמונה בעמ' 10). בשלב זה כבר מופיעה במסמך של eliad ההודעה שהלקוח laptop כתוב hello