

HIT - מכון טכנולוגי חולון  
המחלקה למדעי המחשב

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

קורס רשתות תקשורת מחשבים (0061305-03)

**מגישים:** אלמוג שורצברג

אליעד ביצ'ר

**מרצה:** ד"ר אנדרי קוזוכוב

סמסטר א', תשפ"ו  
ינואר 2026

## תוכן עניינים

2	חלק 1: ניתוח תעבורת פאקטות בפרוטוקול TCP/IP
2	שלב 1 - הכנת 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 התקנה והרצה:
7	דוגמה להרצת היישום:
9	מבנה הקוד - קובץ client.py:
10	מבנה הקוד - קובץ server.py:
11	ניתוח תעבורת רשת ביישום

## חלק 1: ניתוח תעבורת פאקטות בפרוטוקול TCP/IP

בחלק זה נמיר הודעות בשכבת היישום לפקטות TCP/IP ונלכוד את התעבורה שלהן ב-Wireshark. המטרה היא להבין האם הפקטות בכלל נקלטות, משודרות בצורה תקינה ונקלטות במלואן. בהמשך, נבין איך הן נראות.

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

### שלב 1 – הכנת קובץ CSV

בשלב זה הכנו קובץ CSV עם הודעות משכבת האפליקציה שמדמות תעבורה בפרוטוקול HTTP בעזרת בינה מלאכותית Gemini (מצורף בתמונה למטה).

המאפיינים של כל הודעה הם (משמאל לימין): מספר סידורי, סוג הפרוטוקול, מקור, יעד, פורט של המקור, פורט של היעד, תוכן ההודעה, יחידת 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 הנתונים (Encapsulation)

השתמשנו במחברת ה-Jupyter על מנת לארוז את ההודעות לפקטות TCP/IP.

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

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

3. לבסוף הקוד מחבר את הכל כולל ה-payload לכדי פקטת 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 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)

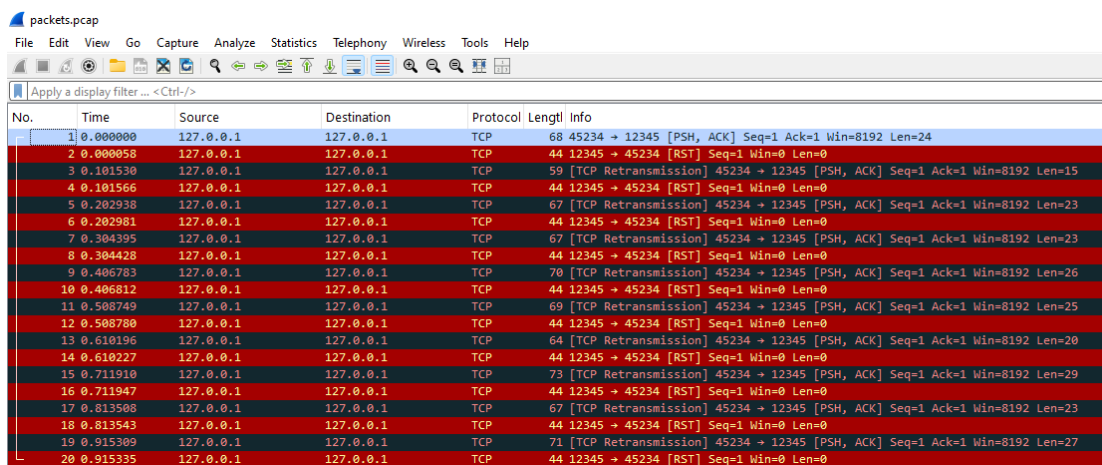
## שלב 3 – לכידה ב-Wireshark

כעת כשיש לנו את הפקטות מוכנות, שלחנו אותן כדי לראות אותן ב-Wireshark ולנתח:

1. באמצעות Wireshark האזנו לממשק Loopback (היעד והמקור הם אותו המחשב ולכן השימוש בממשק ה-Loopback).

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

בתמונה למטה ניתן לראות את הפקטות שלכדנו 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	0.101530	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	0.101566	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	0.711910	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 שכתוב בעמודה INFO בחלון ה WIRESHARK), כלומר הוא מציין בפני הלקוח כי הפורט 12345 סגור ואין אף process במחשב שמאזין לפורט הזה ומורה לו להפסיק את הניסיונות לשלוח את החבילה.

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

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

לולאה זו חוזרת על עצמה בשורות 5-20 ב WIRESHARK. הלקוח מתקדם בהודעות ומסורב פעם אחר פעם ע"י המחשב בגלל שהפורט 12345 סגור. בגלל שמחברת jupyter לא מנהלת את קידום ה sequence number בכל הודעה חדשה שנשלחת, WIRESHARK מניחה כי עצם העובדה שהמספר נשאר 1 מהווה שידור חוזר של הודעות (מה שבפועל לא נכון).

## חלק 2: פיתוח יישום צ'אט

### הסבר כללי על המערכת:

היישום מבוסס על ארכיטקטורת שרת-לקוח שבו השרת הוא רכיב עם כתובת IP קבועה ש"תמיד פועל", כלומר שמחכה ומאזין לפקודות של לקוחות. הלקוחות הם שיזמים את החיבור מולו. התקשורת ביניהם מתבצעת עם TCP Sockets שמתפקדים כממשק בין שכבת האפליקציה לשכבת התעבורה.

הסיבה לשימוש בפרוטוקול TCP ביישום זה היא הצורך בהעברת נתונים אמינה. כלומר אנחנו נדרשים להבטיח שכל ההודעות של הלקוחות יגיעו ליעד, בסדר הנכון ומבלי לאבד מידע.

היישום מחולק לשני קבצי קוד בשפת פייתון: server.py ו-client.py.

## הוראות התקנה והרצה:

1. וודאו שסביבת פייתון מותקנת על המחשבים בהם ירוצו השרת והלקוחות.
2. התקינו את הקבצים server.py ו-client.py. לאחר התקנה עליהם להימצא באותה תיקייה.
3. **לא חובה** - כדי להריץ את הלקוחות והשרת על **מחשבים שונים באותה רשת** עליכם לשנות את ערכי הקבועים SERVER\_IP ו SERVER\_PORT בקובץ server.py לכתובת ה IP הפרטית של המחשב שיריץ את השרת ולפורט פנוי כלשהו. בקובץ client.py יש להתאים בדיוק 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 isnt in use by another ser-  
vice -----##  
SERVER_IP = "localhost"  
SERVER_PORT = 12000
```

4. לחצו מקש ימני על הקובץ server.py ובחרו באופציית copy as path.
5. פתחו חלון CMD במחשב שיריץ את השרת. בחלון זה יש להדביק את הטקסט בלחיצה על CTRL+V ומיד ללחוץ ENTER. הקוד של השרת ירוץ והשרת ידפיס הודעה המאשרת שהוא במצב ONLINE וממתין לחיבורי לקוחות.
6. עבור לקוח חדש: פתחו חלון CMD חדש במחשב שיריץ את הלקוח.
7. לחצו מקש ימני על הקובץ client.py ובחרו באופציית copy as path. יש להדביק בCMD את הטקסט בלחיצה על CTRL+V ומיד ללחוץ ENTER.
8. הקוד של הלקוח ירוץ ויבוצע חיבור לשרת. לאחר הצלחת החיבור, הלקוח יתבקש להזין שם משתמש. יש להזין שם משתמש ייחודי לכל לקוח בעת החיבור. **חזרו על צעדים 6-7 אם ברצונכם להריץ מספר לקוחות.**
9. לשליחת הודעות בין לקוחות יש להשתמש בפורמט הטקסטואלי <message> @target . לדוגמה, אם ברצוננו לשלוח Hey dude למשתמש Bob נרשום @Bob Hey dude .

## דוגמה להרצת היישום:

נריץ את server.py ונקבל את ההודעה הבאה שמשמעותה שהשרת פועל ומאזין לחיבורים:

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

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

```
Run server x
:
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

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

נבחר בשם bob, ונקבל פלט מתאים בחלון client ועדכון בשרת כי נוצר משתמש חדש:

```
server x
:
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'.

client x
:
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 עדכון בשרת:

```
server x
:
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'.

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.py
[SYSTEM] Connected to localhost:12000
Enter your one-word username: alice
[SYSTEM] WELCOME alice. To chat, type: @target <message>
```



כעת bob שלח הודעה אל alice שתוכנה: Hey, how are you doing today? באמצעות הפורמט הטקסטואלי. ההודעה תירשם לוג גם בשרת:

```
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 127.0.0.1:61145 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\PyCharmMiscProject\client.py
[SYSTEM] Connected to localhost:12000
Enter your one-word username: bob
[SYSTEM] WELCOME bob. To chat, type: @
@alice Hey, how are you doing today?
[SYSTEM] WELCOME alice. To chat, type: @target <message>
Message from bob: Hey, how are you doing today?
```

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

```
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 127.0.0.1:61145 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\PyCharmMiscProject\client.py
[SYSTEM] Connected to localhost:12000
Enter your one-word username: alice
[SYSTEM] WELCOME alice. To chat, type: @
Message from bob: Hey, how are you doing today?
exit
[SYSTEM] Disconnected from server.

Command Prompt
C:\Users\almog>
```

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

```
Run
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, isnt 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 $999999999 ?
[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:

מהווה צד הלקוח של היישום. הלקוח יחם חיבור לכתובת השרת (בעזרת קבועים SERVER\_IP ו-PORT שמגדירים לפני הרצה).

כשהוא מתחבר, הלקוח נכנס ללולאה שבה הוא מתבקש להזין שם משתמש כדי להשתמש ביישום. הלולאה תתבצע עד לקבלת אישור מהשרת (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 ללקוח נוצר thread האזנה (פונקציית listen\_for\_messages) במקביל לthread המרכזי שבו הלקוח מתבקש להזין הודעות כדי לצ'ט עם לקוחות אחרים שמחוברים ליישום.

פונקציית listen\_for\_messages: הלקוח נכנס ללולאת האזנה לשרת. כשמתקבלת תגובה מהשרת, 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, צד הלקוח יודע להתמודד עם ניתוקים לא צפויים ויעדכן את הלקוח ואת השרת על כך בזמן אמת.

## מבנה הקוד - קובץ server.py:

מהווה צד השרת של היישום. הוא יוצר Socket מסוג SOCK\_STREAM (כלומר שימוש בפרוטוקול TCP), מבצע bind לפורט ספציפי ומאזין לבקשות נכנסות. צד השרת בנוי על מודל ה-Multi-threading: עבור כל לקוח חדש שמתחבר, השרת יוצר thread נפרד שמקבל את ה-Socket של הלקוח ונשלח להריץ את פונקציית handle\_client\_input.

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

מיד לאחר מכן השרת נכנס שוב ללולאת האזנה לאותו לקוח ספציפי. השרת ממתין להודעות שהלקוח מעוניין לשלוח למשתמשים אחרים שמחוברים גם הם. כשמתקבלת הודעה מהמוען, השרת מחפש את שמו של הנמען (target\_name) במילון ושולח בsocket שלו את ההודעה (כלומר בפועל השרת הוא המתווך בין הלקוחות - לקוח A שירצה לשלוח הודעה ללקוח B, למעשה ישלח את ההודעה לשרת, והשרת הוא זה שישלח את ההודעה 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"]

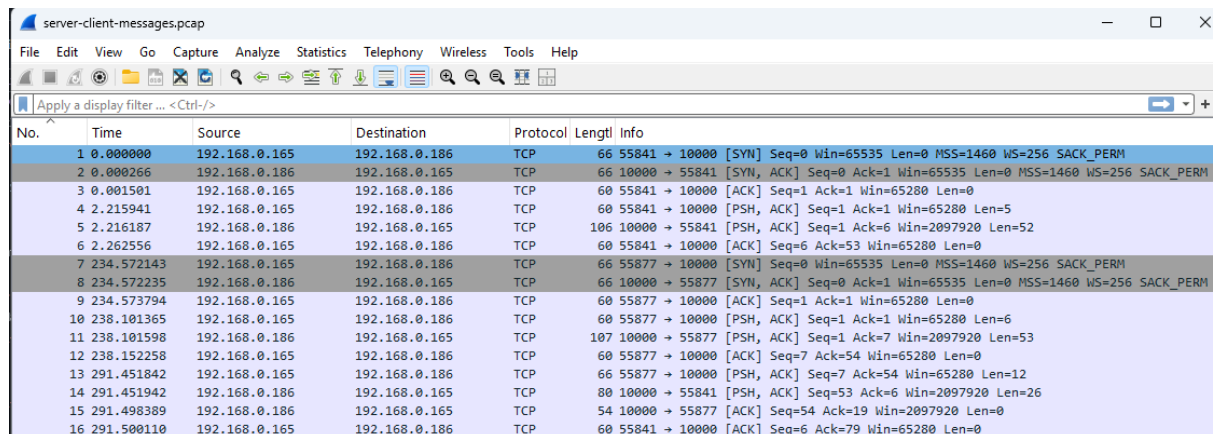
            target_socket.sendall(f"Message from {username}: {message_content}".encode("utf-8"))
            print(f"[LOG] {username} ({formatted_addr}) -> {target_name} ({target_ip}): {message_content}")
        else:
            client_socket.sendall("WRONG_USAGE".encode("utf-8"))
```

לאורך כל התכנית, צד השרת יודע להתמודד עם ניתוקים לא צפויים ויעדכן על כך בזמן אמת.

## ניתוח תעבורת רשת ביישום

לצורך בחינת התעבורה בפרויקט, הפעלנו שרת ממחשב אחד ו 2 לקוחות ממחשב אחר. שני המחשבים פועלים באותה הרשת. את לכידת הפקטות ביצענו באמצעות Wireshark ושמרנו לתוך קובץ server-client-messages.pcap המצורף לפרויקט.

נסביר את התעבורה לפי שורות שממוספרות בעמודה No. בתמונה למטה:



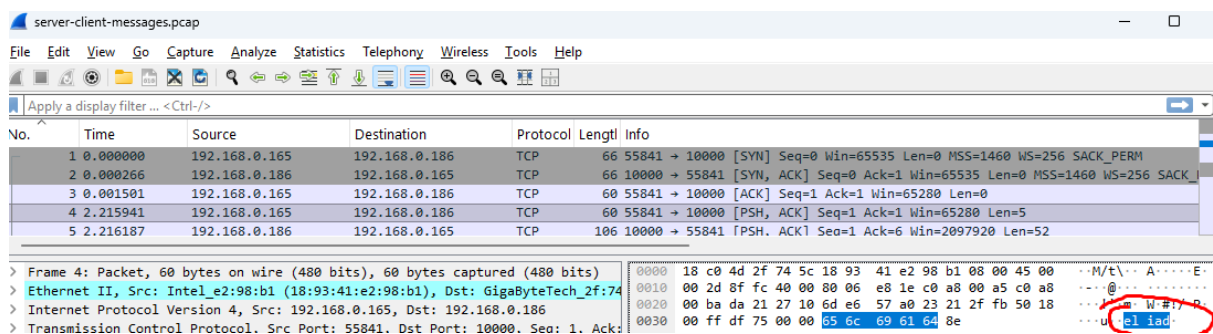
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.165	192.168.0.186	TCP	66	55841 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000266	192.168.0.186	192.168.0.165	TCP	66	10000 → 55841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3	0.001501	192.168.0.165	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: מתחיל TCP Handshake. לקוח ראשון יחם חיבור לשרת (שכבר פועל וממתין לחיבורים). הלקוח שולח פקטה עם דגל SYN מופעל. משורה זו מבינים כי ה IP של הלקוח והפורט שלו הם 192.168.0.165:55841 (זהו ה SOURCE) וכי ה IP של השרת הוא 192.168.0.186 שמאזין על פורט 10000 (זהו ה DESTINATION).

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

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

שורה 4: הלקוח שולח 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) on interface 0  
> Ethernet II, Src: Intel\_e2:98:b1 (18:93:41:e2:98:b1), Dst: GigaByteTech\_2f:74:00:00:00:00  
> 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, Win: 65280, Len: 5

שורה 5: השרת שולח הודעת WELCOME שאורכה Len=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

> Frame 5: Packet, 106 bytes on wire (848 bits), 106 bytes captured (848 bits)	0000	18 93 41 e2 98 b1 18 c0	4d 2f 74 5c 08 00 45 00	..A.... M/t\..E..
> Ethernet II, Src: GigaByteTech_2f:74:5c (18:c0:4d:2f:74:5c), Dst: Intel_e2:98:b1:18:c0:4d:2f:74:5c	0010	00 5c 59 24 40 00 80 06	1e c8 c0 a8 00 ba c0 a8	..Y\$@.....
> Internet Protocol Version 4, Src: 192.168.0.186, Dst: 192.168.0.165	0020	00 a5 27 10 da 21 23 21	2f fb 6d e6 57 a5 50 18	...!#/..m..W.P..
> Transmission Control Protocol, Src Port: 10000, Dst Port: 55841, Seq: 1, Ack: 1, Win: 65280, Len: 5	0030	20 03 c7 7b 00 00 57 45	4c 43 4f 4d 45 20 65 6c	...ME LCOME e1
> Data (52 bytes)	0040	69 61 64 2e 20 54 6f 20	63 68 61 74 2c 20 74 79	iad. To chat, ty
	0050	70 65 3a 20 40 74 61 72	67 65 74 5f 6e 61 6d 65	pe: @tar get name
	0060	20 3c 6d 65 73 73 61 67	65 3e	<messag e>

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

שורה 7: מאותו מחשב של הלקוח eliad הרצנו לקוח חדש שיזם חיבור לשרת (ההוכחה לכך - SOURCE זהה לSOURCE שיזם חיבור לשרת בשורה 1). הפורט של הלקוח החדש הוא 55877. מתחיל תהליך נוסף של TCP Handshake.

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

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

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

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	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)	0000	18 c0 4d 2f 74 5c 18 93	41 e2 98 b1 08 00 45 00	..M/t\.. A.....E..
> Ethernet II, Src: Intel_e2:98:b1 (18:93:41:e2:98:b1), Dst: GigaByteTech_2f:74:5c (18:c0:4d:2f:74:5c)	0010	00 2e 90 00 40 00 80 06	e8 19 c0 a8 00 a5 c0 a8	..@.....
> Internet Protocol Version 4, Src: 192.168.0.165, Dst: 192.168.0.186	0020	00 ba da 45 27 10 c6 c1	24 dd 03 5a 21 c0 50 18	...E'... \$...Z!..P..
> Transmission Control Protocol, Src Port: 55877, Dst Port: 10000, Seq: 1, Ack: 1, Win: 65280, Len: 6	0030	00 ff cd c2 00 00 6c 61	70 74 6f 70	.....la ptop

שורה 11: השרת שולח הודעת WELCOME לlaptop שאורכה Len=53.

שורה 12: הלקוח laptop מאשר (ACK) שקיבל את הודעת ה WELCOME של השרת.

שורה 13: הלקוח laptop רוצה לשלוח הודעה באורך Len=12 ללקוח eliad שהתוכן שלה הוא hello. הוא מעביר את ההודעה לשרת שמשמש כמתווך. זיהינו שמדובר בלקוח laptop מכיוון שהפורט של המחשב השולח את הפקאט 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	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
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

```

> 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:74:5c (18:c0:4d:2f:74:5c)
> 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: 54, Len: 12
Data (12 bytes)
0000 18 c0 4d 2f 74 5c 18 93 41 e2 98 b1 08 00 45 00  ..M/t\...E...
0010 00 34 90 02 40 00 00 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...e liad hel
0040 6c 6f                                     lc

```

שורה 14: השרת מעביר את ההודעה ללקוח בפורט 55841, כלומר ללקוח eliad. אורך ההודעה גדל 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	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
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

```

> Frame 14: Packet, 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
> Ethernet II, Src: GigaByteTech_2f:74:5c (18:c0:4d:2f:74:5c), Dst: Intel_e2:98:b1 (18:93:41:e2:98:b1)
> 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: 6, Len: 26
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 00 06 1e df c0 a8 00 ba c0 a8  ..BY'@...
0020 00 a5 27 10 da 21 23 21 30 2f 6d e6 57 a5 50 18  ...!#! 0/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 lapto p: hello

```

שורה 15: השרת מאשר (ACK) ללקוח laptop (מזהים שזה הלקוח laptop לפי הפורט 55877 שבתמונה בעמ' 10) שהוא קיבל את תוכן ההודעה שרוצים להעביר ללקוח eliad (בפועל השרת כבר העביר את ההודעה ע"פ שורה 14 ושולח אישור באיחור).

שורה 16: הלקוח eliad מאשר (ACK) שקיבל את ההודעה שהשרת העביר לו (מזהים שזה הלקוח eliad לפי הפורט 55841 שבתמונה בעמ' 10). בשלב זה כבר מופיעה במסך של eliad ההודעה hello שהלקוח laptop כתב.