

# דוח מסכם: פרויקט גמר ברשתות תקשורת מחשבים

## ומימוש יישום צ'אט TCP/IP נושא: ניתוח תעבורה בפרוטוקול

מגישות:

רום אשכנזי ת.ז. 214438186

עדן שראיזין ת.ז. 324929223

קישור ל-github: <https://github.com/litttttttttttt/project.git>

### 1. חלק ראשון - אריזה ולכידת מנות:

#### א. יצירת קובץ הנתונים

בשלב הראשון יצרנו קובץ נתונים בפורמט CSV המתאר רצף הודעות בשכבת היישום (Application Layer) בפרוטוקול HTTP מעל TCP.

כל שורה בקובץ מייצגת הודעה לוגית בין לקוח (client\_browser) לשרת (web\_server), וכוללת מזהה הודעה (msg\_id), פרוטוקול היישום (app\_protocol), צד מקור ויעד (src\_app, dst\_app), תוכן ההודעה (message) וזמן שליחה יחסי (timestamp).

תהליך היצירה בוצע באופן ידני בשילוב כלי AI (Chat GPT) כדי לייצר תבנית שיחה אמינה בין שרת ללקוח.

#### קובץ ה-CSV:

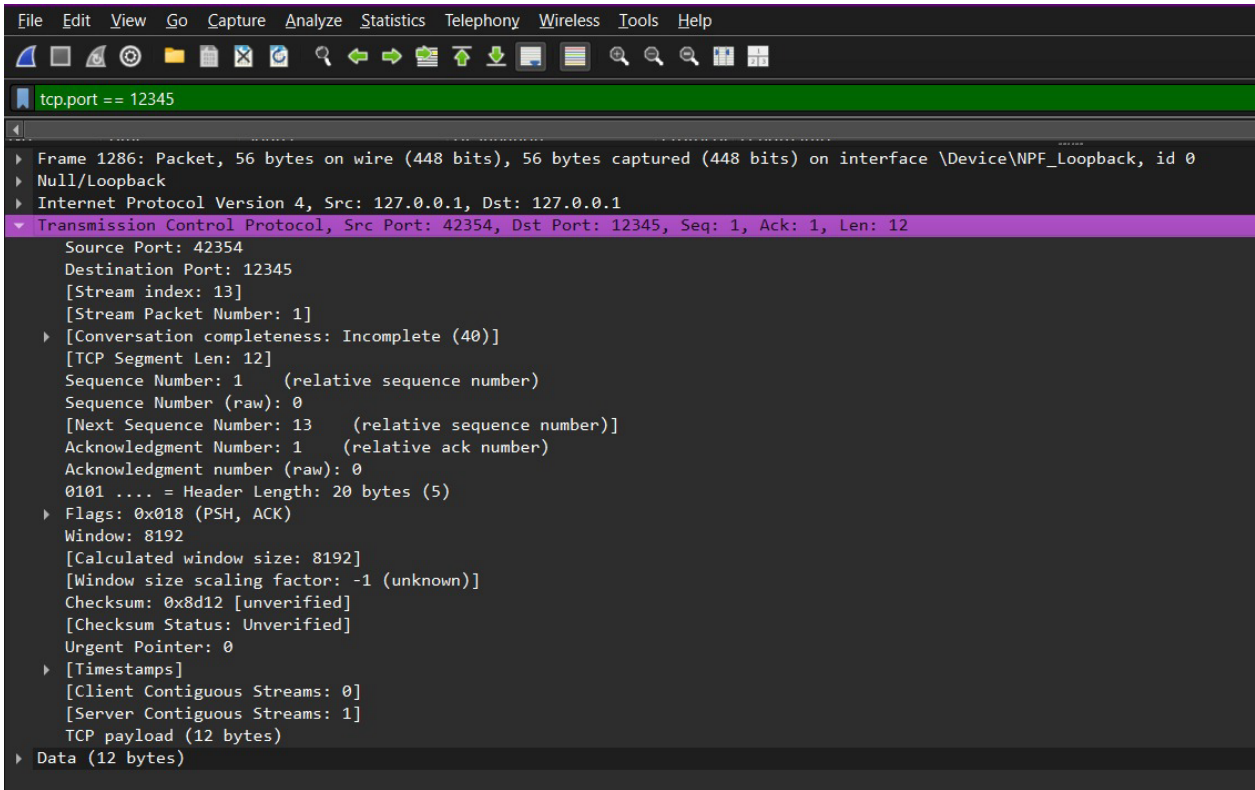
msg_id	app_protocol	src_app	dst_app	message	timestamp
1	HTTP	client_browser	web_server	GET /index.html	0.015
2	HTTP	web_server	client_browser	200 OK (image/png)	0.020
3	HTTP	client_browser	web_server	GET /styles.css	0.025
4	HTTP	web_server	client_browser	200 OK (text/css)	0.030
5	HTTP	client_browser	web_server	POST /login	0.045

## ב. תהליך האריזה (Encapsulation Process)

השתמשנו במחברת Jupyter ובקוד Python כדי לדמות את תהליך האריזה במערכת ההפעלה:

1. **שכבת היישום**: התוכנה קראה את ההודעה מה-CSV
2. **שכבת התעבורה (Transport)**: הקוד הוסיף כותרת TCP (כולל מספרי פורטים וגלים).
3. **שכבת הרשת (Network)**: ההודעה נשלחת דרך כתובת ה-loopback (127.0.0.1), וניתן לצפות במנות ה-IP שנוצרות ב-Wireshark.

בפלט המחברת ניתן לראות את המידע הבינארי (Hexdump) שמראה איך ההודעה "עטופה" בכותרות השונות לפני השליחה, ה-Hexdump מדגים כיצד ההודעה משכבת היישום נעטפת בהדרגה בכותרות השונות לפני השליחה.



```
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 12345
Frame 1286: Packet, 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{...}, id 0
  Null/Loopback
  Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  Transmission Control Protocol, Src Port: 42354, Dst Port: 12345, Seq: 1, Ack: 1, Len: 12
    Source Port: 42354
    Destination Port: 12345
    [Stream index: 13]
    [Stream Packet Number: 1]
    [Conversation completeness: Incomplete (40)]
    [TCP Segment Len: 12]
    Sequence Number: 1 (relative sequence number)
    Sequence Number (raw): 0
    [Next Sequence Number: 13 (relative sequence number)]
    Acknowledgment Number: 1 (relative ack number)
    Acknowledgment number (raw): 0
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x018 (PSH, ACK)
    Window: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x8d12 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    [Timestamps]
    [Client Contiguous Streams: 0]
    [Server Contiguous Streams: 1]
    TCP payload (12 bytes)
  Data (12 bytes)
```

## ג. תהליך הלכידה וניתוח ב-Wireshark

לצורך ניתוח התעבורה הופעלה תוכנת Wireshark על ממשק ה loopback, בזמן שליחת הודעות בין הלקוח לשרת.

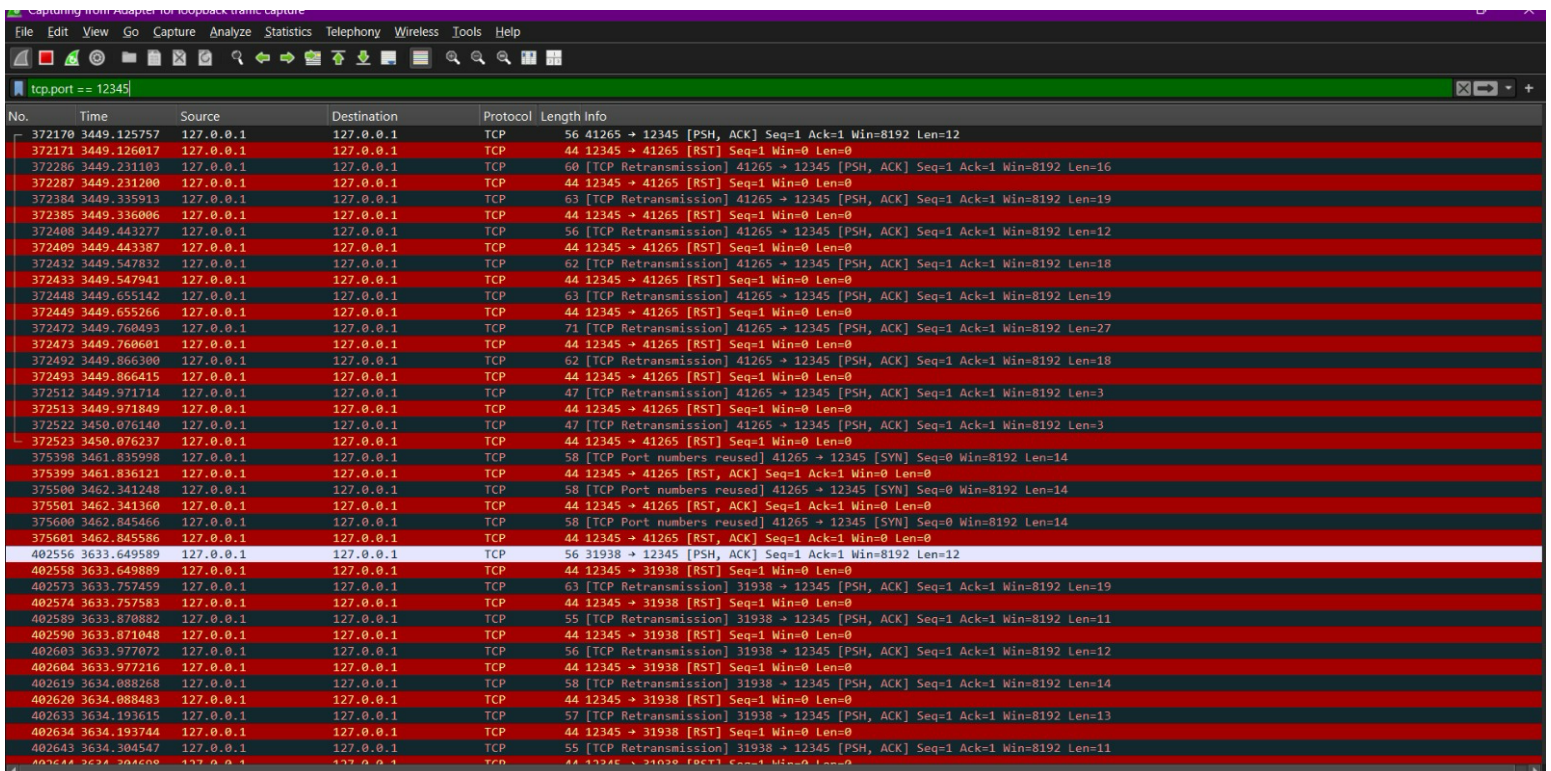
בלכידה ניתן לראות תעבורת TCP כאשר כתובת המקור והיעד הן 127.0.0.1 והתקשורת מתבצעת מול פורט השרת 12345.

בנוסף, ניתן להבחין כי הדגל PSH מציין כי הנתונים שב-TCP payload נשלחים מיידית ליישום בצד המקבל.

## ד. תיאור והסבר תעבורה שנלכדה ב-Wireshark:

במהלך הרצת המערכת נלכדה תעבורת רשת באמצעות Wireshark על ממשק ה-loopback (127.0.0.1). התעבורה שנצפתה מייצגת זרימת תקשורת TCP בין לקוח לשרת מקומיים, כאשר החיבור מתבצע דרך פורט היעד 12345.

בתמונה מוצגת זרימת תעבורת TCP הכוללת רצף של חבילות שנלכדו ב-Wireshark, המועברות דרך ממשק ה-loopback (127.0.0.1) בין לקוח לשרת מקומיים בפורט 12345, כאשר חלק מהחבילות מכילות payload המייצג נתונים שנשלחו משכבת היישום, רצף החבילות והמאפיינים שלהן מדגימים את אופן העברת המידע בין שכבות המערכת ואת תהליך התקשורת בין הלקוח לשרת.



No.	Time	Source	Destination	Protocol	Length	Info
372170	3449.125757	127.0.0.1	127.0.0.1	TCP	56	41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=12
372171	3449.126017	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372286	3449.231103	127.0.0.1	127.0.0.1	TCP	60	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=16
372287	3449.231200	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372384	3449.335913	127.0.0.1	127.0.0.1	TCP	63	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=19
372385	3449.336006	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372408	3449.443277	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=12
372409	3449.443387	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372432	3449.547832	127.0.0.1	127.0.0.1	TCP	62	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=18
372433	3449.547941	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372448	3449.655142	127.0.0.1	127.0.0.1	TCP	63	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=19
372449	3449.655266	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372472	3449.760493	127.0.0.1	127.0.0.1	TCP	71	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=27
372473	3449.760601	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372492	3449.866300	127.0.0.1	127.0.0.1	TCP	62	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=18
372493	3449.866415	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372512	3449.971714	127.0.0.1	127.0.0.1	TCP	47	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=3
372513	3449.971849	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
372522	3450.076140	127.0.0.1	127.0.0.1	TCP	47	[TCP Retransmission] 41265 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=3
372523	3450.076237	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST] Seq=1 Win=0 Len=0
375398	3461.835998	127.0.0.1	127.0.0.1	TCP	58	[TCP Port numbers reused] 41265 → 12345 [SYN] Seq=0 Win=8192 Len=14
375399	3461.836121	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
375500	3462.341248	127.0.0.1	127.0.0.1	TCP	58	[TCP Port numbers reused] 41265 → 12345 [SYN] Seq=0 Win=8192 Len=14
375501	3462.341360	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
375600	3462.845466	127.0.0.1	127.0.0.1	TCP	58	[TCP Port numbers reused] 41265 → 12345 [SYN] Seq=0 Win=8192 Len=14
375601	3462.845586	127.0.0.1	127.0.0.1	TCP	44	12345 → 41265 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
402556	3633.649589	127.0.0.1	127.0.0.1	TCP	56	31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=12
402558	3633.649889	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402573	3633.757459	127.0.0.1	127.0.0.1	TCP	63	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=19
402574	3633.757583	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402589	3633.870882	127.0.0.1	127.0.0.1	TCP	55	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11
402590	3633.871048	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402603	3633.977072	127.0.0.1	127.0.0.1	TCP	56	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=12
402604	3633.977216	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402619	3634.088268	127.0.0.1	127.0.0.1	TCP	58	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14
402620	3634.088483	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402633	3634.193615	127.0.0.1	127.0.0.1	TCP	57	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=13
402634	3634.193744	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0
402643	3634.304547	127.0.0.1	127.0.0.1	TCP	55	[TCP Retransmission] 31938 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11
402644	3634.304608	127.0.0.1	127.0.0.1	TCP	44	12345 → 31938 [RST] Seq=1 Win=0 Len=0

## חלק 2: יצירת יישום וניתוח תעבורה

### א. הסבר כללי על מערכת ומבנה הקוד

המערכת שנבנתה מדמה תקשורת ברשת באמצעות פרוטוקול TCP והיא מורכבת משרת ולקוחות.

השרת אחראי על קבלת חיבורים מלקוחות, תיווך בין הודעות וניהול רשימת הלקוחות הפעילים.

הלקוחות מתחברים לשרת, שולחים אליו הודעות ומקבלים הודעות שנשלחו על ידי לקוחות אחרים דרך השרת.

המטרה של המערכת היא לאפשר העברת הודעות בין לקוחות, כאשר כל ההודעות עוברות דרך השרת שמשמש כמתווך.

כל לקוח מזוהה לפי שם ייחודי שנשלח בתחילת ההתחברות, מה שמאפשר לשרת לעקוב אחרי מי שלח כל הודעה ולאן להעביר אותה.

המימוש נעשה באמצעות ספריית Socket של פייתון לצורך תקשורת ובאמצעות ספריית Threading להפעלת שרת שתומך במספר חיבורים במקביל (5 לקוחות בו זמנית).

מבנה הקוד מתחלק לשני רכיבים עיקריים:

☐ שרת (Server) - מאזין לבקשות התחברות, יוצר תהליך (Thread) נפרד עבור כל לקוח, מקבל ממנו הודעות ומעביר אותן ללקוחות אחרים.

☐ לקוח (Client) - מתחבר לשרת, שולח את שמו, שולח הודעות ומאזין להודעות שמגיעות מהשרת.

השרת שומר מילון של לקוחות מחוברים, שבו כל לקוח מזוהה לפי השם שהוא שלח ולכל שם מקושר החיבור הפעיל. כאשר הודעה מתקבלת מלקוח, השרת מנתב אותה ללקוחות הרלוונטיים לפי לוגיקה פנימית.

### ב. הוראות התקנה והרצה

#### הרצת השרת-

- ☐ יש להריץ את הקובץ server.py.
- ☐ השרת מאזין לחיבורים נכנסים על פורט ברירת מחדל (לדוגמה 12345)
- ☐ השרת תומך בחיבור של 5 לקוחות במקביל.

#### הרצת הלקוח-

- ☐ כל לקוח מופעל על ידי הרצת הקובץ client.py.
- ☐ עם ההרצה, המשתמש מתבקש להזין שם ייחודי.
- ☐ ניתן לפתוח מספר מופעים של לקוח (בחלונות נפרדים) לצורך בדיקת תקשורת בין מספר משתמשים.

## ג. דוגמאות קלט ופלט:

□ בתמונה זו מוצגת הרצת הקובץ `server.py`, השרת נכנס למצב האזנה וממתין לחיבורים מלקוחות:

```
Anaconda Prompt - python s  × + v

(base) C:\Users\Eden>conda activate networks

(networks) C:\Users\Eden>cd Desktop

(networks) C:\Users\Eden\Desktop>dir
Volume in drive C has no label.
Volume Serial Number is 8207-228D

Directory of C:\Users\Eden\Desktop

18/12/2025  12:59    <DIR>          .
14/12/2025  11:49    <DIR>          ..
18/12/2025  12:53    <DIR>          .ipynb_checkpoints
18/12/2025  12:59             1,538 client.py
18/11/2024  07:22             2,371 Excel.lnk
14/12/2025  13:07             411 group03_tcpip_project1_input.csv
14/12/2025  12:57             428 group03_tcpip_project_input.csv
14/12/2025  10:56             437 group05_tcpip_project_input.csv
18/11/2024  09:48             2,392 Person 1 - Chrome.lnk
18/11/2024  07:22             2,392 PowerPoint.lnk
14/12/2025  13:09            18,828 raw_tcp_ip_notebook_fallback_annotated-v1.ipynb
18/12/2025  12:57             4,770 server.py
24/07/2025  11:22    <DIR>          study
18/12/2025  12:53              72 Untitled.ipynb
18/11/2024  07:22             2,409 Word.lnk
14/12/2025  13:41           665,565 מנסמ חוד.docx
02/07/2025  10:22             432 יפניא תואחסונן ףד.htm
02/07/2025  10:22           509,305 הדידב תואחסונן ףד.pdf
12/06/2025  10:59           8,069,594 הדידב תורדגה 2.pdf
24/04/2025  21:46           381,233 1 (1).pdf יפניאב (יקלח) מיאשונ מוניס
                16 File(s)          9,662,177 bytes
                4 Dir(s)  906,110,529,536 bytes free

(networks) C:\Users\Eden\Desktop>python server.py
```

□ בתמונה זו מוצגת הרצת לקוח והתחברות- הלקוחה אליס מתחברת לשרת, לאחר ההתחברות מופיעה הודעת ברוכים הבאים ולאחר מכן מתחילות להופיע הודעות מערכת על התחברויות חדשות ושליחת הודעות בין משתמשים.

```
Anaconda Prompt - python c  × + v

(base) C:\Users\Eden>conda activate networks

(networks) C:\Users\Eden>cd Desktop

(networks) C:\Users\Eden\Desktop>python client.py 127.0.0.1 5000 alice
SYS|Welcome. Identify with: HELLO|<your_name>
OK|Registered as alice
SYS|Commands: LIST, SEND|to|msg, QUIT
SYS|bob connected
LIST
USERS|alice,bob
SEND|bob|יה יב
OK|Sent to bob
FROM|bob|יה יב
SYS|carol connected
SYS|dave connected
SYS|erin connected
SEND|erin|קידב 5
OK|Sent to erin
|
```

□ הרצת לקוח נוסף בשם דייב- המשתמש דייב מתחבר לשרת, מקבל את רשימת המשתמשים הקיימים במערכת ושולח הודעה לבוב. ניתן לראות שהתהליך דומה להרצת הלקוח הראשון וכי המערכת תומכת במספר לקוחות בו זמנית.

```
Anaconda Prompt - python c  × + v

(base) C:\Users\Eden>conda activate networks

(networks) C:\Users\Eden>cd Desktop

(networks) C:\Users\Eden\Desktop>python client.py 127.0.0.1 5000 dave
SYS|Welcome. Identify with: HELLO|<your_name>
OK|Registered as dave
SYS|Commands: LIST, SEND|to|msg, QUIT
SYS|erin connected
LIST
USERS|alice,bob,carol,dave,erin
SEND|bob|יה יב
OK|Sent to bob
|
```



## ד. ניתוח תעבורה של היישום עד שכבת הרשת

היישום עושה שימוש בפרוטוקול TCP לצורך תקשורת בין הלקוחות לשרת. כל לקוח יוצר חיבור TCP באמצעות כתובת IP מקומית (127.0.0.1), החיבור נשמר פתוח לאורך זמן ההתקשרות ומאפשר שליחה וקבלה של הודעות טקסטואליות ברמת היישום.

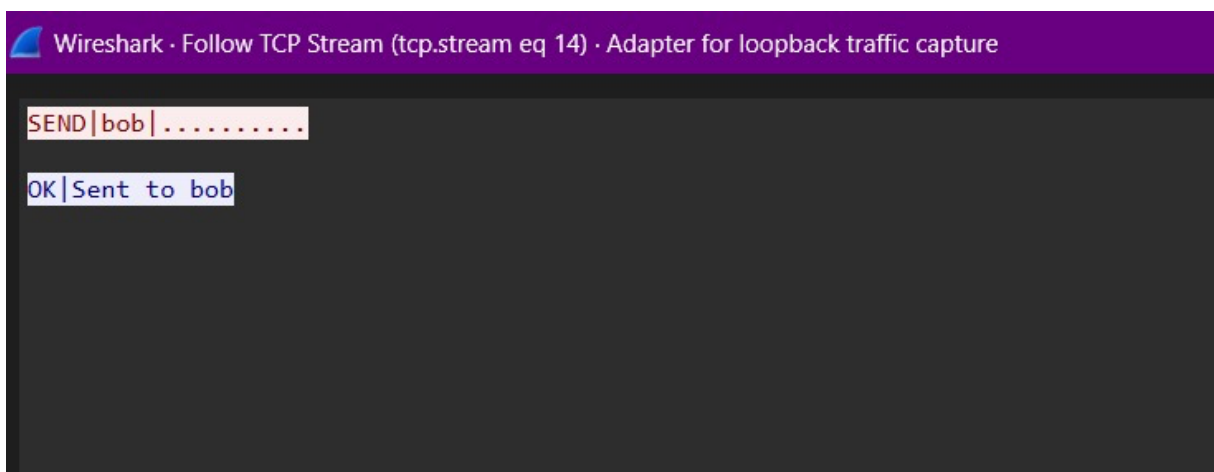
ברמת שכבת היישום (Application layer) – הלקוחות שולחים פקודות בפורמט טקסטואלי כגון:

- ☐ SEND|bob|<message> - שליחת הודעה ללקוח אחר דרך השרת
- ☐ QUIT או LIST כמו נוספות

ב-Wireshark, באמצעות סינון לפי tcp.stream ניתן לראות בבירור את תוכן ההודעות שנשלחו.

בתמונה הראשונה מוצגת הודעת SEND|bob|..... שנשלחה מהלקוח לשרת ולאחריה הודעת התגובה מהשרת

OK|Sent to bob, המעידה על עיבוד מוצלח של הבקשה ברמת היישום.



בתמונה השנייה מוצגת רשימת חבילות ה-TCP הרלוונטיות לאותו stream. ניתן לזהות:

- ☐ חבילות TCP מסוג [PSH, ACK] הכוללות payload, כלומר נתוני יישום.
- ☐ חבילות [ACK] המאשרות קבלת מידע.
- ☐ שימוש ב-sequence number ו-acknowledgment number המעידים על מנגנון האמינות של TCP.

The image shows the Wireshark packet list for "tcp.stream eq 14". The table has columns: No., Time, Source, Destination, Protocol, and Length Info.

No.	Time	Source	Destination	Protocol	Length Info
161	60.979775	127.0.0.1	127.0.0.1	TCP	64 55943 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=20
162	60.979862	127.0.0.1	127.0.0.1	TCP	44 5000 → 55943 [ACK] Seq=1 Ack=21 Win=255 Len=0
165	60.980406	127.0.0.1	127.0.0.1	TCP	59 5000 → 55943 [PSH, ACK] Seq=1 Ack=21 Win=255 Len=15
166	60.980427	127.0.0.1	127.0.0.1	TCP	44 55943 → 5000 [ACK] Seq=21 Ack=16 Win=255 Len=0

ניתוח זה מדגים כיצד פקודות ברמת היישום מתורגמות לתעבורת TCP/IP וכיצד ניתן לעקוב אחר זרימת המידע משכבת היישום ועד שכבת הרשת באמצעות כלי ניתוח תעבורה.

### **חלק 3: תיאור שימוש בבינה מלאכותית**

במהלך העבודה נעשה שימוש בכלי בינה מלאכותית (ChatGPT) ככלי עזר להבנה ופתרון בעיות. המטרה המרכזית הייתה לקבל הבהרות בנוגע לדרישות המשימה, למבנה הרצוי של המערכת, ולהבין טוב יותר את עקרונות העבודה עם sockets ו-TCP בפיתוח. נעשה שימוש בכלי לקבלת עזרה בזיהוי תקלות בקוד ובפתרון בעיות טכניות שעלו במהלך הפיתוח. דוגמאות פרומפטים:

□ "תוכל בבקשה להסביר לי על השלב שבו צריך לנהל חיבורים בין לקוחות, מה השרת אמור לעשות?"

□ "איך אני יכולה לראות ב Wireshark את ההודעה שנשלחה מהלקוח לשרת?"

□ "יש לי קוד שמתחיל לעבוד ואז נתקע, איך אני יכולה לזהות איפה הבעיה?"