

# ניתוח תעבורה ב프וטוקול IP/TCP

פרויקט גמר בקורס רשתות תקשורת מחשבים

סטודנט: יוסי אריאל  
ת.ז: 209227842  
מרצה: אפי פרופו

## תוכן עניינים

### חלק ראשון: אריזת נתונים ולכידת מנות בעזרת Wireshark

דרך יצירת קובץ CSV עם הودעות בשכבה היישום (שלבי עבודה).....	עמוד 4
תהליך והסביר של דרך יצירת מנות .....	עמוד 5-7
תהליך והסביר של תהליך לכידה .....	עמוד 8-9

### חלק שני: כתיבת יישום רשות וניתוח תעבורת של אותו יישום

הסבר כללי על מבנה המערכת.....	עמוד 10
הסבר כללי על מבנה הקוד.....	עמוד 10
הוראות הרצה והפעלה.....	עמוד 11
דוגמאות קלט ופלט.....	עמוד 12-16
ניתוח תעבורת של היישום עד שכבת הרשת (כולל).....	עמוד 17-22

### חלק שלישי: תיאור שימוש בביבנה מלאכותית

מטרות שימוש.....	עמוד 23
דוגמאות לפרומפטים.....	עמוד 23

# אריזת נתונים ולכידת מנות בעזרת Wireshark

דרך ייצירת קובץ CSV עם הودעות בשכבה היישום-שלבי עבודה

## שלב 1: הכנות קובץ CSV

קובץ קלט CSV המכיל הודעות בשכבה היישום ב프וטוקול HTTP נוצר. הקובץ נוצר באמצעות בינה מלאכותית (AI) כפי שנרשם שניתן להשתמש בהוראותuproject.

הקובץ כולל:

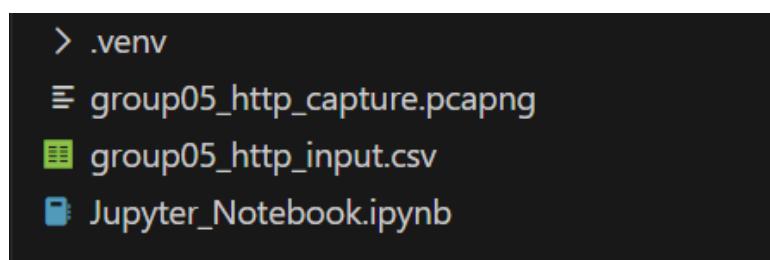
- ❖ מזהה הודעה
- ❖ פרוטוקול שימוש
- ❖ מקור
- ❖ יעד
- ❖ תוכן ההודעה
- ❖ חותמת זמן ייחסית

שם הקובץ: group05\_http\_input.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	HTTP/1.1 200 OK	0.028
3	HTTP	client_browser	web_server	GET /style.css	0.042
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.057
5	HTTP	client_browser	web_server	GET /script.js	0.073
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.089
7	HTTP	client_browser	web_server	POST /login	0.115
8	HTTP	web_server	client_browser	HTTP/1.1 302 Redirect	0.134
9	HTTP	client_browser	web_server	GET /dashboard	0.162
10	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.187

## שלב 2: שימוש במחברת Jupyter

- ❖ המחברת Jupyter נפתחה בצורה תקינה על ידי שימוש ב VS code .
- ❖ קובץ ה CSV נתון בצורה תקינה.
- ❖ שלבי הקוד במחברת הוריצו בצורה תקינה.
- ❖ נוצרה הדמיה ייצרת אריזת נתונים ב TCP .
- ❖ נוצרה תעבורת שתלכד ב Wireshark .

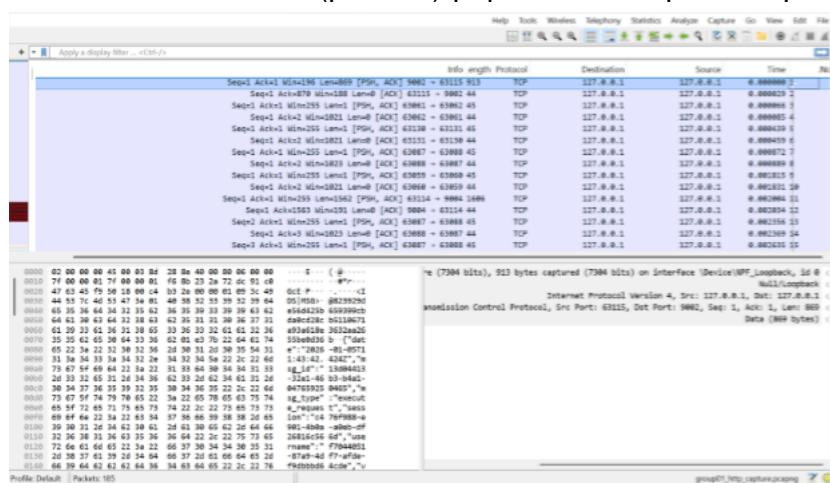


## דרך ייצרת קובץ CSV עם הודעות בשכבה היישום-שלבי עבודה (המשר)

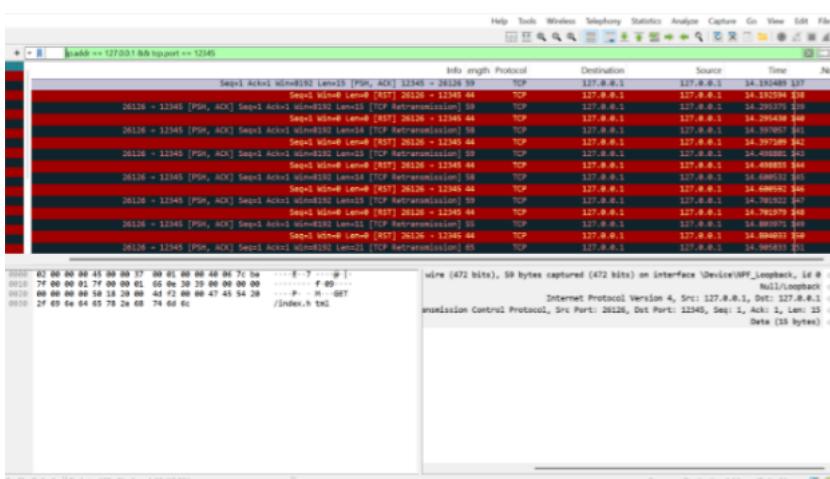
### שלב 3: לכידה ב-Wireshark

- ❖ הופעל Wireshark כראוי.
- ❖ המחברת הריצה וונצ'ר תעבורת b .Wireshark את הלכידה והקובץ נשמר בפורמט pcapng (הקובץ נשמר ללא סינון, הוא נשמר במלואו) ולאחר מכן ניתן לסקן מבוקש).
- ❖ הסימולציה ריצה על localhost ובפורט 12345, וכן התעבורת הרלוונטית לא עברה דרך פורט 80 (פורט 80 לרלוונטי לתעבורת HTTP רגילה באינטרנט).
- ❖ לכן, נדרש ניתוח השימושי במסנן: `ip.addr == 127.0.0.1 && tcp.port == 12345` כפ' שנקtab במחברת jupyter

להלן צילום מסך של שבירת הקובץ (לא מסנן):



להלן צילום מסך של שבירת הקובץ (עם מסנן):



## תיאור והסבר של תהליך אריזת מנות

נתאר את תהליך אריזת המנות (הפקות) לפי שלבים ונראה תיאור של הפרטים באמצעות wireshark.  
בקובץ CSV כל שורה שבו מייצגת הודעה בשכבה היחסום.  
כפי שלמדנו, כאשר היא נשלחת היא עוברת תהליכי של אריזת מנות.  
לצורך הדגמת השלבים ופירוט באמצעות wireshark נבחרה באופן רנדומלי פקטה מס' 141 לצורך המחשה והדגמה.

### **שלב ראשון:**

קיימת רק הודעה בעמודת message בCSV

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	HTTP/1.1 200 OK	0.028
3	HTTP	client_browser	web_server	GET /style.css	0.042
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.057
5	HTTP	client_browser	web_server	GET /script.js	0.073
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.089
7	HTTP	client_browser	web_server	POST /login	0.115
8	HTTP	web_server	client_browser	HTTP/1.1 302 Redirect	0.134
9	HTTP	client_browser	web_server	GET /dashboard	0.162
10	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.187

### **שלב שני:**

אותן ההודעות נארחות בTCP ומשתלבים להן פורט מקור ופורט יעד.  
(בפרויקט הפורט יעד הוא 12345)

ניתן לראות את הפרטים בTransmission Control Protocol של אותה פקטה 141:

- ❖ פורט מקור: 26126
- ❖ פורט יעד: 12345
- ❖ הדגמים ACK ו-SNK ACK המסמנים העברת נתונים דרך TCP.
- ❖ ACK=1 הנתונים התקבלו בצורה תקינה בצד המקבל.

## תיאור והסבר של תהליך אריזת מנות (המשר)

### שלב שלישי:

قطع ה TCP נארץ בתווך חבילת IP (כליום, נארץ בשכבות האינטרנט).

No	Time	Source	Destination	Protocol	Length	Info
137	14.192489	127.0.0.1	127.0.0.1	TCP	59	Seq=1 Ack=1 Win=15 [PSH, ACK] 12345 + 26126
138	14.192594	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 [RST] 26126 + 12345
139	14.295375	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=192 Len=15 [TCP Retransmission]
140	14.295438	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
145	14.600532	127.0.0.1	127.0.0.1	TCP	58	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
147	14.781922	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
148	14.701979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
149	14.803971	127.0.0.1	127.0.0.1	TCP	55	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission]
150	14.804033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345
151	14.905833	127.0.0.1	127.0.0.1	TCP	65	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission]

Frame 141: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF\_Loopback, id 0 < Null/Loopback <  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 <  
Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14 <

כאשר נסתכל ב协议 Internet, ניתן לראות:

- ❖ כתובות מקור: 127.0.0.1
- ❖ כתובותיעד: 127.0.0.1

★ הפרויקט מבוצע על מחשב יחיד ולכן הכתובת מקור זהה לכתובתיעד.

## תיאור והסבר של תהליך אריזת מנות (המשר)

### שלב רביעי:

אריזת נתונים בשכבות התקשורת.

No	Time	Source	Destination	Protocol	Length	Info
137	14.192489	127.0.0.1	127.0.0.1	TCP	59	Seq=1 Ack=1 Win=8192 Len=15 [PSH, ACK] 12345 + 26126 59
138	14.192594	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
139	14.295375	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission] 59
140	14.295430	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission] 58
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission] 59
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
145	14.600532	127.0.0.1	127.0.0.1	TCP	58	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission] 58
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
147	14.781922	127.0.0.1	127.0.0.1	TCP	59	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission] 59
148	14.781979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
149	14.883971	127.0.0.1	127.0.0.1	TCP	55	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission] 55
150	14.884033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 + 12345 44
151	14.985833	127.0.0.1	127.0.0.1	TCP	65	26126 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission] 65

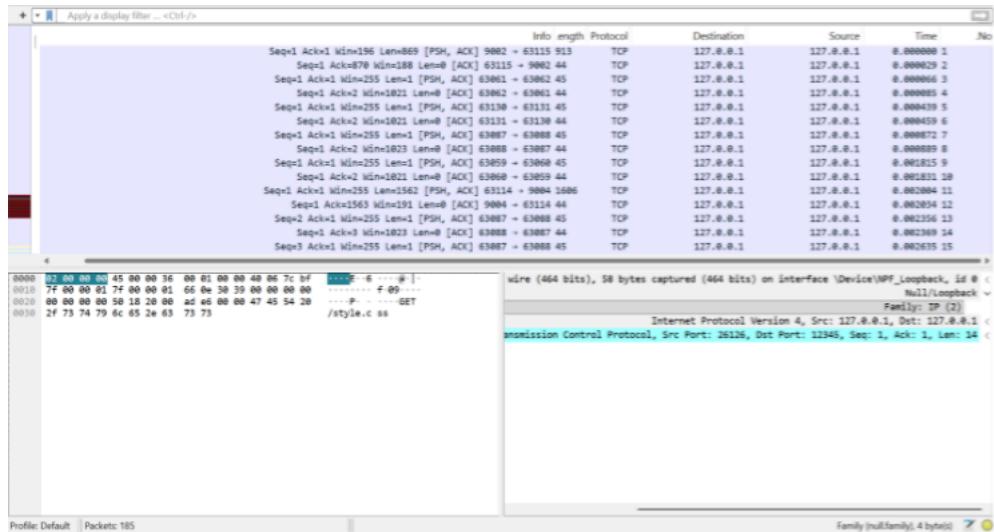
Frame 141: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF\_Loopback, id 0 < Null/Loopback <  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 <  
Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14 <

בחולנית המידע עבור פקטה 141, ניתן לראות שמודיע בשורה הראשונה Frame 141:

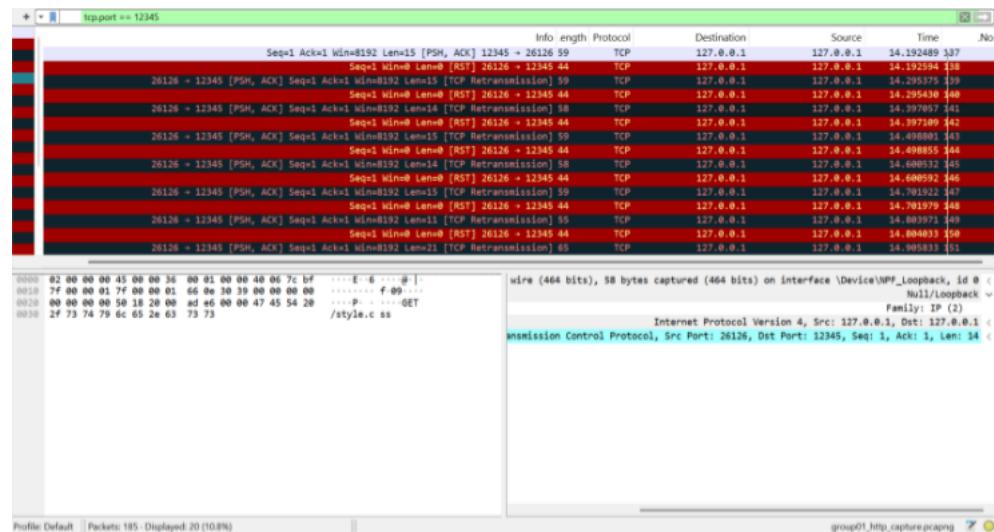
- ❖ מס' מס' פקטה 141.
- ❖ הגודל שהוא נלכדה 58 בתים.

## תיאור והסבר של תהליך הילכידה

כאשר הקוד במחברת jupyter הורץ, הופעלת ייחודי מערכת wireshark לפני ההווראות.  
 התקשרות בפרויקט מתבצעת על כתובת 127.0.0.1  
 ניתן לראות באופן כללי שאכן יש פקודות (185 פקודות) ואכן מתבצעת תעבורת:

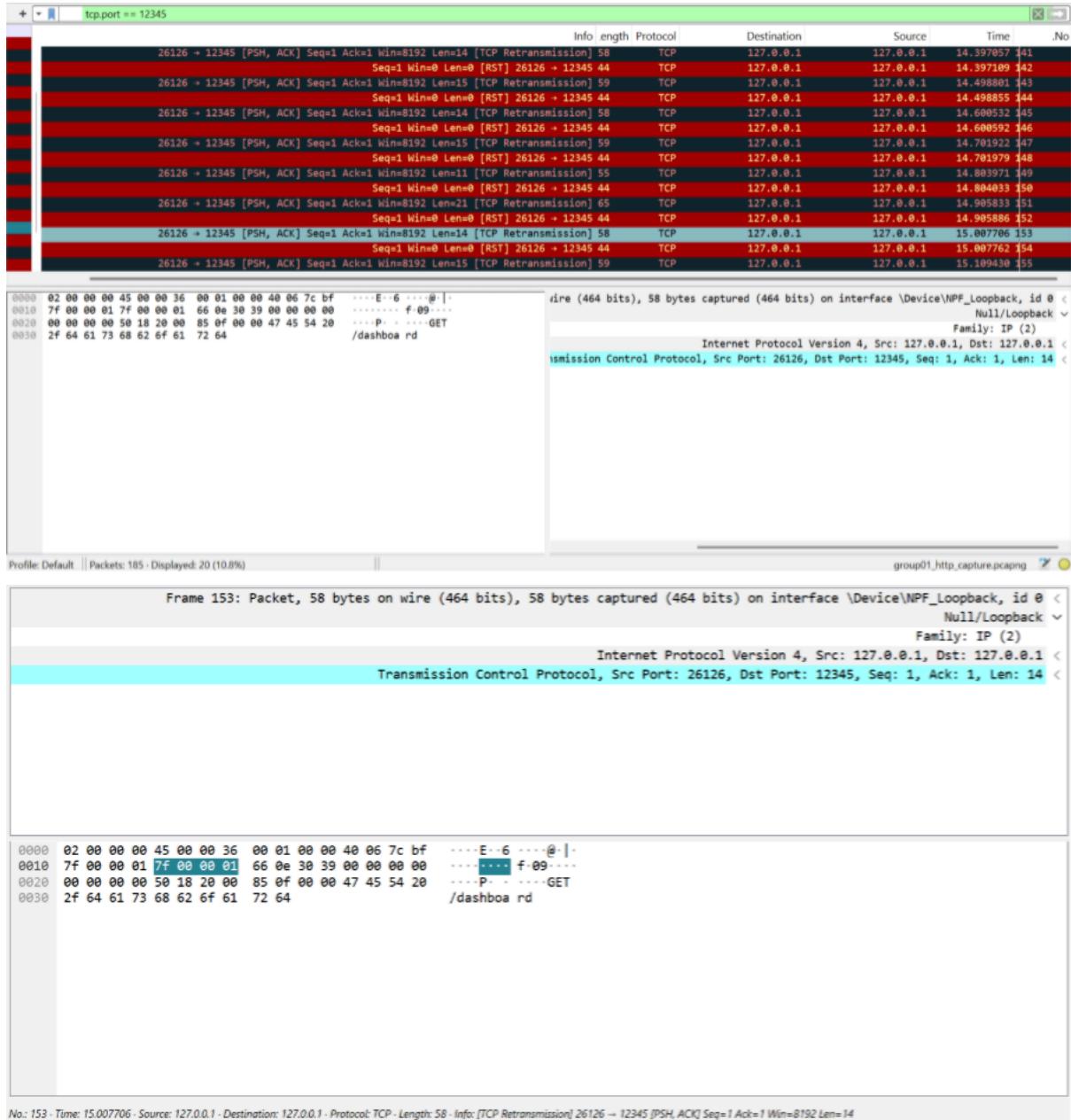


.(TCP) לאחר שלכדה תעבורה, הורץ שלב 5 במחברת jupyter שיצר תעבורה TCP, נוכל לראות כמה פקודות TCP tcp.port==12345 ואת מספר הפורט שאכן תואם לקוד שהורץ:



## תיאור והסבר של תהליך הילכידה (המשר)

לבסוף, לאחר שהילכידה נעצרה (הקובץ גם נשמר) ניתן לבחור פקטה ולבצע ניתוח עמוק של שכבות התקשרות שהיא עברה (כפי שהראיתי קודם לכן).  
באופן כללי, אבחר באופן רנדומלי פקטה מס' 153 ונוכל לקבל את הפרטים המתאים:



# כתיבת יישום רשות וניהוט תעבורה של אותו יישום

## הסבר כללי על המערכת

בחלק השני של הפרויקט, נבנתה מערכת המדממת שרת-לקוח על פי פרוטוקול TCP. המערכת מאפשרת למספר משתמשים (לקוחות) להתחבר לשרת מרכזי שמתווך ביניהם. המערכת מאפשרת יצירת צ'אטים נפרדים בין המשתמשים ובכך ניתן לשלוח ולקבל הודעות פרטיות ממשתמשים ספציפיים.

כל זה מתבצע בזמן אמיתי בו ניתן לשמור את המשתמשים (לקוחות) שמתחברים בצד של השרת, השתמשתי בנתונים מסווג מילון כדי לשמור את המשתמשים (לקוחות) לשרת. לכל משתמש יש מצבה ספציפי.

מבנה נתונים מסווג מילון מאפשר גישה מהירה למשתמשים וגם ניהול וניהוט נוחים בין המשתמשים.

- ❖ הרשות מודעת לחיבורים כניסה באמצעות פורט קבוע (5000), ומנהל את כל החיבורים באמצעות sockets.
- ❖ כל לקוח מתחבר לשרת עם שם משתמש שהוא בוחר, והרשות משמש כמתווך בין המשתמשים ואינו משתמש בשיחות עצמן.
- ❖ המערכת תומכת במספר המשתמשים המתחברים בו זמנית, ומתפלת במקרי ניתוק לא צפויים. (במקרה של ניתוק משתמש, הרשות מעדכן את רשימת המשתמשים המתחברים, מעדכן עלusr, והלקוח מציג הודעה מערכת מתאימה בצ'אט הפטרי).
- ❖ הודעות נשלחות באופן פרטי בלבד בין שני משתמשים נבחרים, ואין משודרות לשאר המשתמשים המתחברים למערכת.
- ❖ התקשרות מתבצעת על גבי IP (localhost (127.0.0.1)) שמאפשר עובודה נוחה עם המערכת.
- ❖ הצד הלקוח קיים ממשק גרפי (GUI) שמאפשר עובודה נוחה עם המערכת.

## הסבר כללי על מבנה הקוד

המערכת מחולקת לשני קבצי קוד עיקריים.

1. צד שרת (yk.server) שאחראי על החיבורם, ניהול המשתמשים והכמות, וניהוט ההודעות.
2. צד לקוח (yk.client) שאחראי על חיבור לשרת, שליחת וקבלת הודעות (בנוסף גם קיימים הצד לקוח מימוש למשק GUI שנוח יותר למשתמש).

▼ COMPUTER\_NETWORK\_PROJECT\_PART2

client.py

server.py

## הוראות הרצה והפעלה

ראשית, הפרויקט נבנה בסביבה עובדה: `python visual studio code` בשפת `python`.  
לכן, עדיפות להפעיל את המערךת גם בסביבה עובדה `visual studio code` עם תמייה ב `python` או בסביבה  
עובדת אחרת התומכת ב `python`.

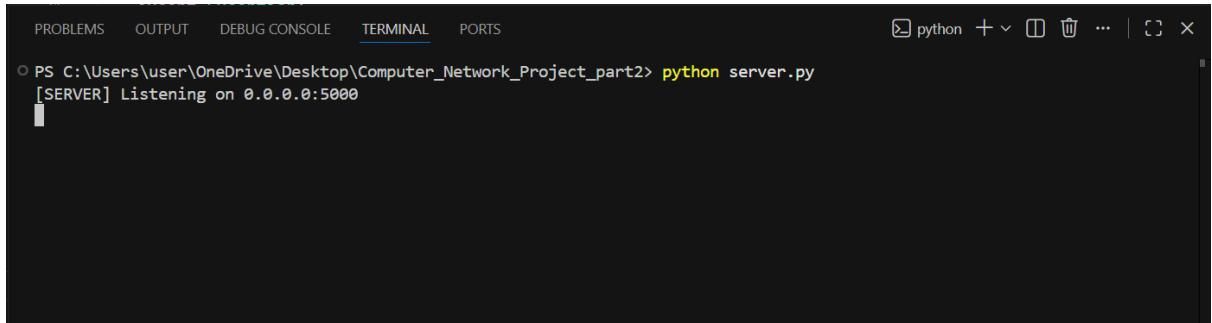
1. יש לפתח את תיקיית הפרויקט בסביבת הפיתוח.
2. יש להריץ את קובץ השרת `server.py`:  
פתיחת terminal new terminal.  
לאחר מכן, יש להקליד "python server.py" (כموון ללא גרשימים).
3. יש להריץ את קובץ הלוקה `client.py`:  
פתיחת terminal new terminal.  
לאחר מכן, יש להקליד "python client.py" (כמוון ללא גרשימים).
4. לאחר הרצת קובץ הלוקה ייפתח חלון GUI המאפשר למשתמש להזין שם משתמש ולהתחבר לשרת.  
כאשר תקליד שם משתמש ותלחץ "connect" - תראה את שם המשתמש בסרגל בצד.
5. ניתן להריץ מספר פעמים את קובץ הלוקה `uk.py` (לפי פתיחת alink new terminal כפי שמתואר בסעיף 3) ולפתח מספר משתמשים בו זמנית עם שמות משתמשים שונים.  
כל המשתמשים המוחברים יופיעו בסרגל בצד.
6. השרת יציג מידע על חיבורים וניתוקים של משתמשים, כולל פרטי החיבור (כתובת IP ומספר פורט).
7. לאחר התחברות המשתמשים, ניתן לבחור משתמש מהרשימה ולהתחליל שיחה פרטית.  
ההודעות נשלחות באופן פרטיא בין שני המשתמשים בלבד, ואשר המשתמשים אינם יכולים לראות או לגשת אליהן.

### הערה חשובה:

המערכת פועלת על כתובת localhost ובפורט קבוע, ולכן אינה דורשת הגדרות רשות מיוחדות.  
במידת הצורך, יש להגיד:  
❖ כתובת שרת 127.0.0.1  
❖ פורט 5000

## דוגמאות קלט ופלט

❖ **חיבור שרת (server)**  
נ裏ץ ב terminal  
python server.py

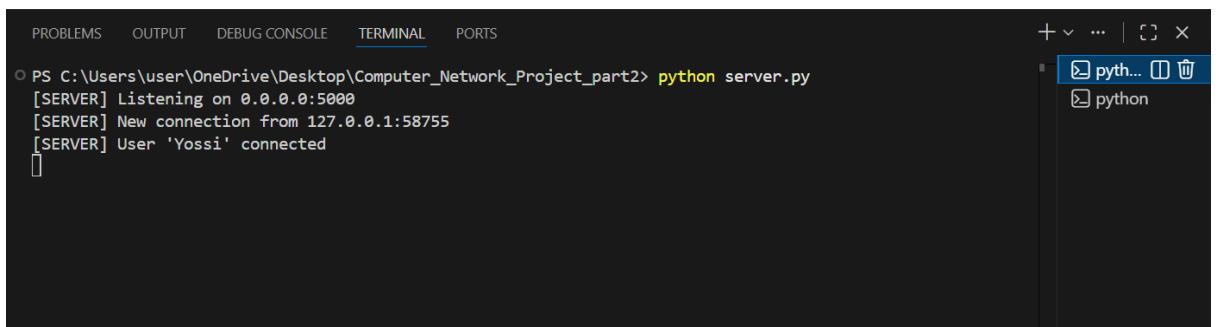
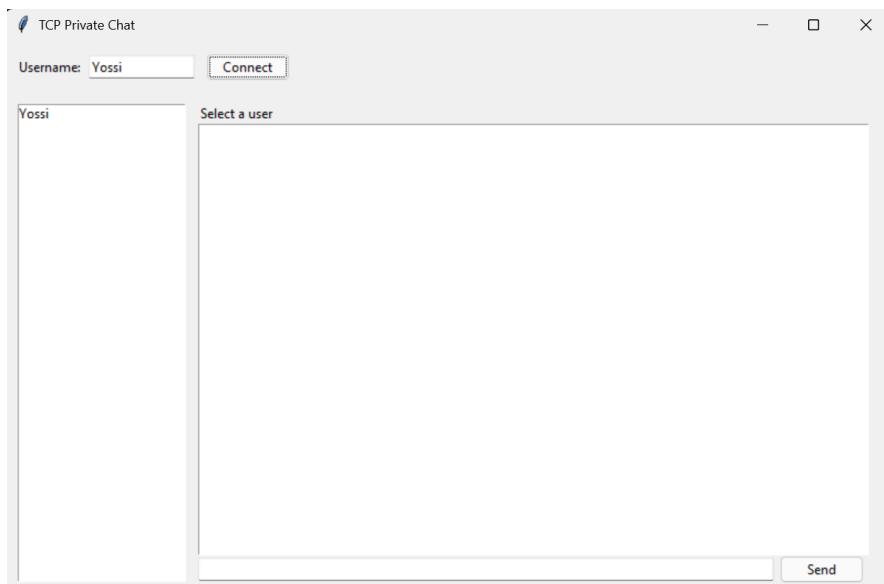


A screenshot of a terminal window titled "python". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal content shows:

```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
```

נקבל הודעה שהשרת אכן התחבר + פרטי חיבור (כתובת IP ומספר פורט).

❖ **חיבור משתמש (client)**  
נ裏ץ ב terminal חדש terminal  
נקליד שם משתמש ונתחבר.  
נראה בחלוןית GUI את שם המשתמש הראשון שהתחבר מופיע בסרגל בצד.  
ובחלוןית השרת ניתן לראות את פרטי ההתחברות שלו (כתובת IP ומספר פורט) ואת שם המשתמש.



A screenshot of a terminal window titled "python". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The terminal content shows:

```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python client.py
[CLIENT] Connected to 127.0.0.1:5000
```

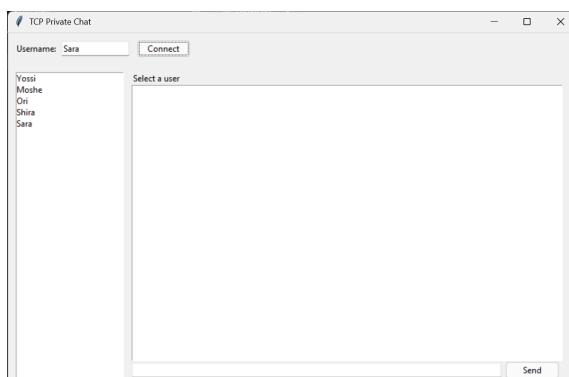
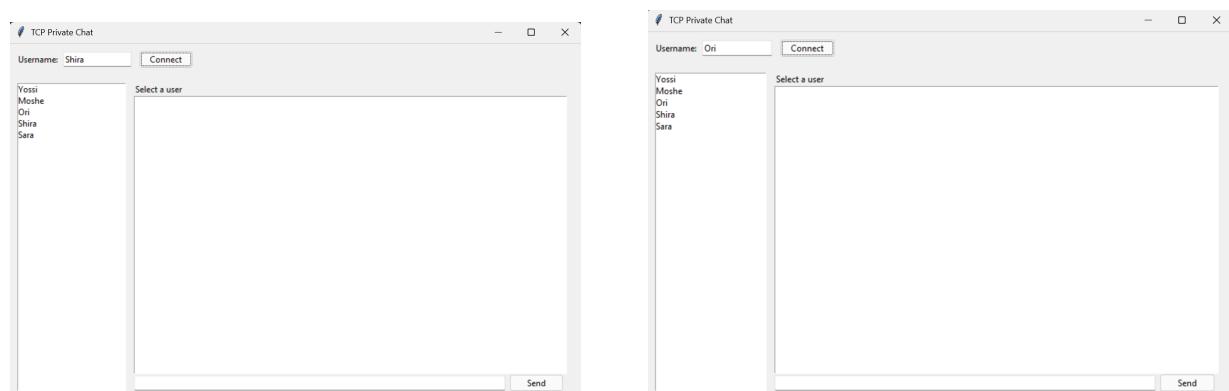
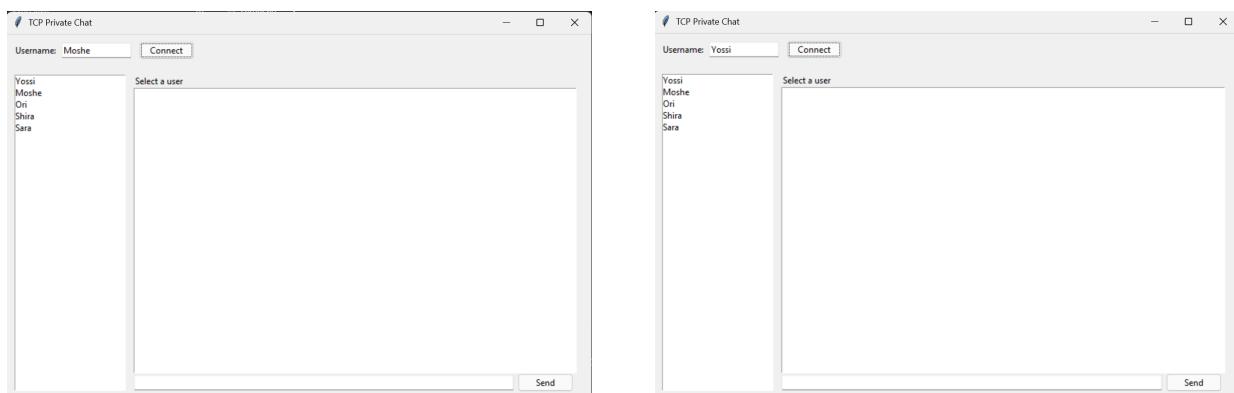
To the right of the terminal, there is a sidebar with a list of open terminals:

- python
- python

## דוגמאות קלט ופלט (המשך)

### ❖ חיבור 5 משתמשים בו זמנית

5 מחוברים בו זמנית התחברו לשרת: "Yossi" , "Moshe" , "Ori" , "Shira" , "Sara"  
כל שמות המשתמשים מופיעים בסרגל צד בחלונית GUI ובנוסף השרת עדכן על כל החיבורים ופרט  
החיבור שלהם (כתובת IP ומספר פורט).

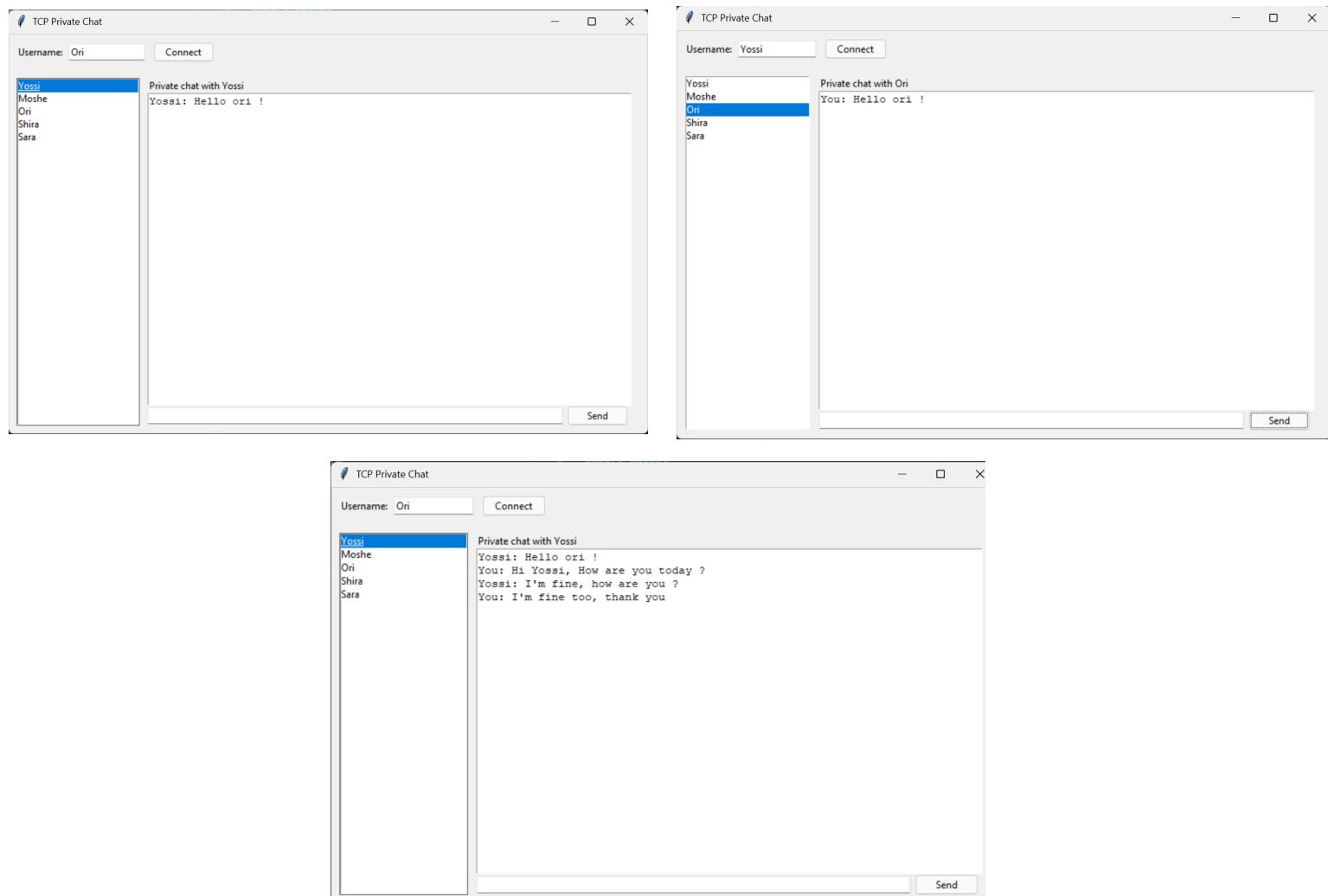


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:58755
[SERVER] User 'Yossi' connected
[SERVER] New connection from 127.0.0.1:58818
[SERVER] User 'Moshe' connected
[SERVER] New connection from 127.0.0.1:58821
[SERVER] User 'Ori' connected
[SERVER] New connection from 127.0.0.1:58824
[SERVER] User 'Shira' connected
[SERVER] New connection from 127.0.0.1:58827
[SERVER] User 'Sara' connected
```

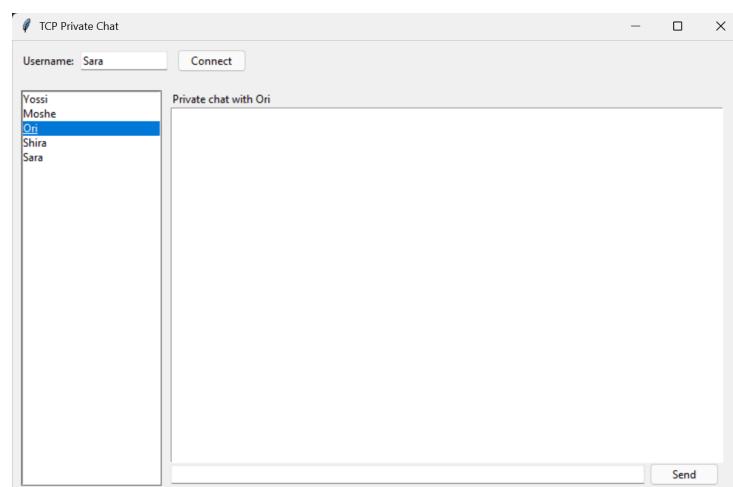
## דוגמאות קלט ופלט (המשך)

### ❖ פתיות צ'אט פרטי ושליחת הודעה

לצורך הדוגמה, המשתמש "Yossi" בוחר את המשתמש "Ori" ושולח לו הודעה.  
לאחר מכן, מתנהל ביןיהם שיחה פרטית שלא גלויה לשאר המשתמשים.

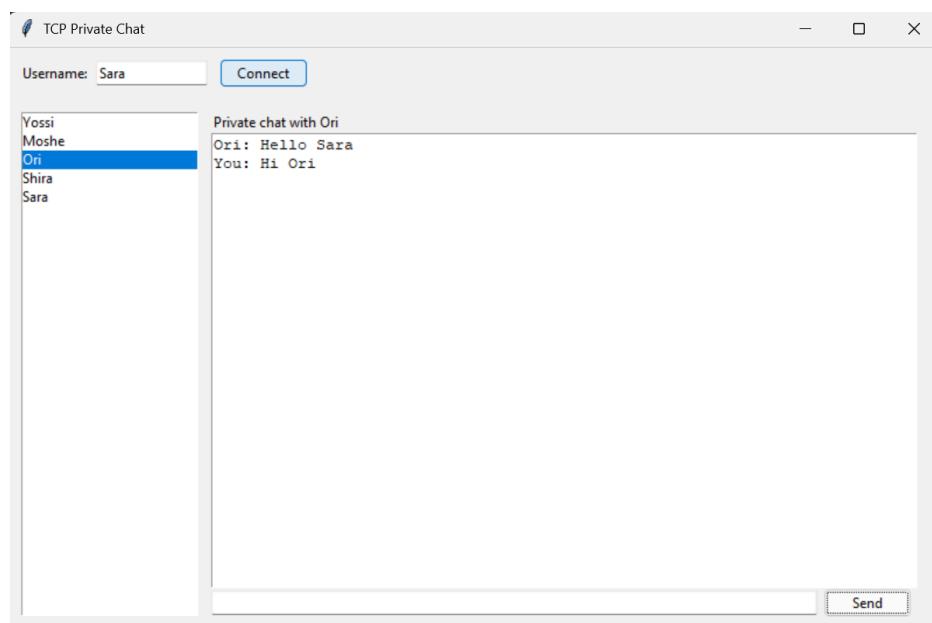
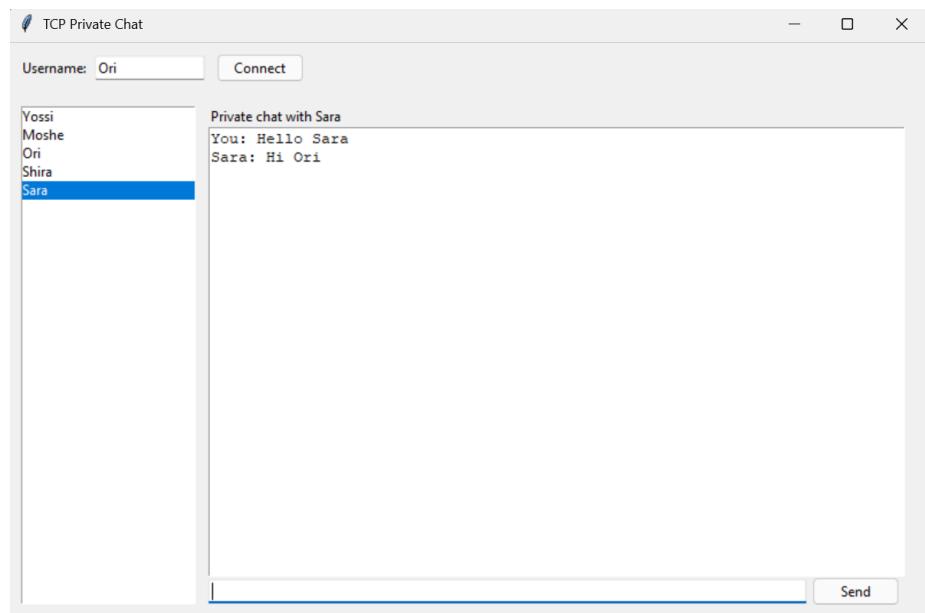


נראה גם שהצ'אט אכן פרטי ומستخدم אחר (לצורך הדוגמא "Ori" ו "Shira") לא יכול לראותו:



## דוגמאות קלט ופלט (המשך)

- ❖ **פתיחה צ'אט נוסף בין משתמשים תוך כדי צ'אט אחר עם משתמש שונה.**  
לצורך הדוגמה, ניקח את "Ori" וננהל שיחת עם "Sara" בזמן שיש לו שיחה עם "Yossi".

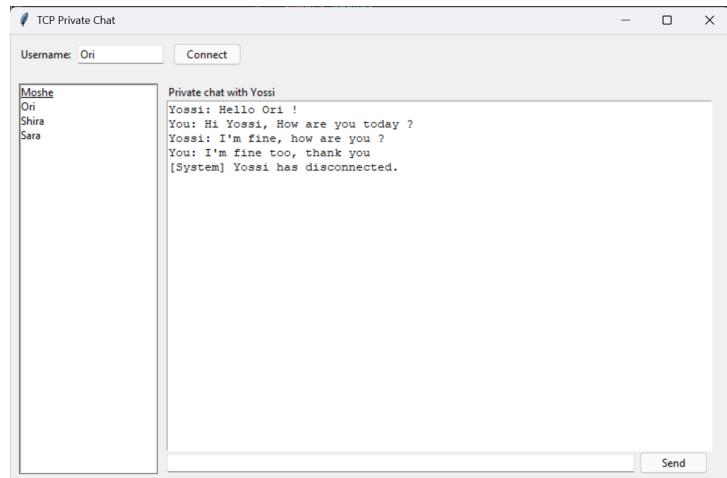


## דוגמאות קלט ופלט (המשר)

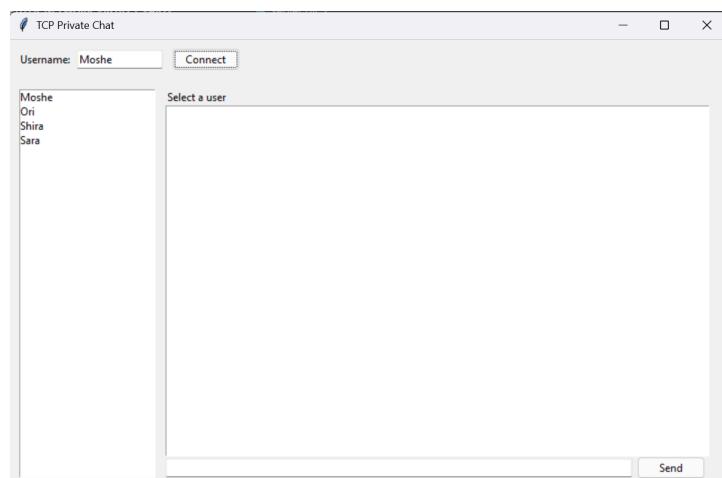
### ❖ ניתוק משתמש מהמערכת

בஹמך לדוגמא בו ה"יטה שיחה בין "Ori" ו "Yossi" נקבע ניתוק של "Yossi" מהמערכת.

נראה שמתקובלת הודעה לאורי על ניתוק של "Yossi".  
וגם ניתן לראות שהוא ירד בסרגל בצד (הוא ירד כל המשתמשים).



אצל משתמש אחר (למשל "Moshe") ניתן לראות ש "Yossi" לא מופיע גם בסרגל צד לאחר הניתוק שלו.



```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:63357
[SERVER] User 'Yossi' connected
[SERVER] New connection from 127.0.0.1:63360
[SERVER] User 'Moshe' connected
[SERVER] New connection from 127.0.0.1:63366
[SERVER] User 'Ori' connected
[SERVER] New connection from 127.0.0.1:63369
[SERVER] User 'Shira' connected
[SERVER] New connection from 127.0.0.1:63372
[SERVER] User 'Sara' connected
[SERVER] User 'Yossi' disconnected
```

## ניתוח תעבורת היפר-טראנספורם עד שכבת הרשת (כולל)

### שליחת הודעה ממוגנת ספציפי למשתמש אחר ספציפי:

ראשית, פתחתי את הhypertext wireshark על מצב Npcap Loopback Adapter והחלה את לכידת התעבורה. לאחר מכן, עברתי ל-VS code והרמתי את המ מערכת תקן.

ניתן לציין לצורך הדגמה, פתחתי שוב 5 משתמשים שונים בשםות: "Yossi", "Moshe", "Ori", "Shira" ו-Sara

"Hello, my name is Yossi" שלחת הودעה מהמשתמש "Yossi" למשתמש ספציפי "Shira" ההודעה היא: "Hello Yossi".

לאחר מכן, ביצעתו ניתוק של המשתמש "Yossi" מהמערכת.

לבסוף, חזרתי בחזרה ל-wireshark, עכברתי את הלכידה, ופילטרתי בфиילטר מתאים על מנת לראות את הפיקוטות המתאימות.

### שכבת היפר-טראנספורם:

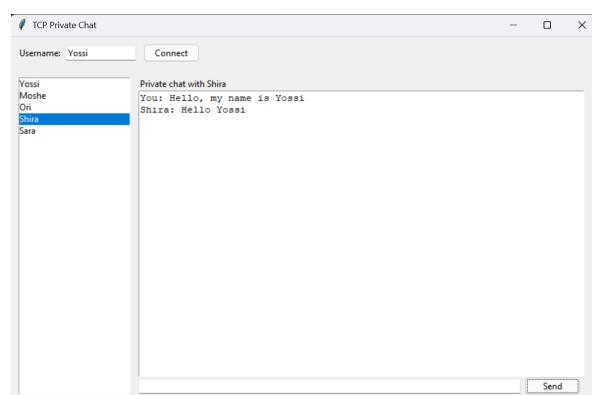
בפרויקט, שכבת היפר-טראנספורם ממומשת בצד הלקוח ובצד השרת.

בצד הלקוח, נוצר צ'אט בטלפון GUI שדרכו המשתמש יכול לשולח את ההודעות.

בצד השרת, מtbody קבלת ההודעה, ניתוח ההודעה ושליחתה לשאר המשתמשים במערכת.

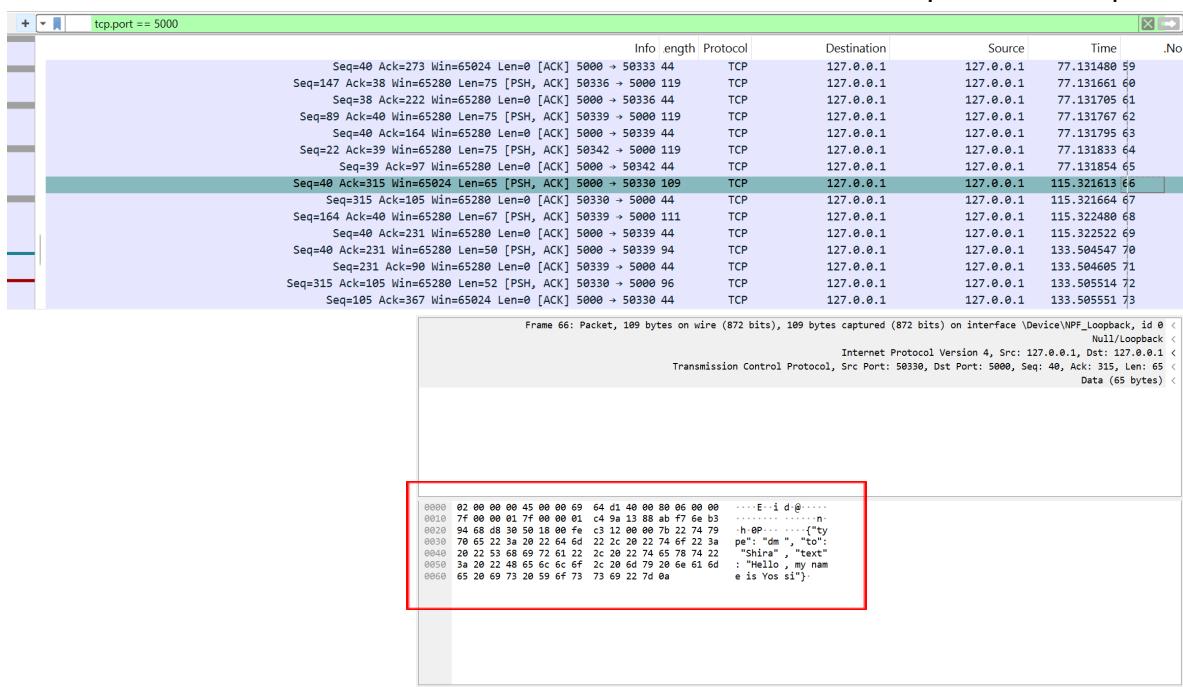
נוצרה הודעה ממוגנת ספציפי "Yossi" בצד, ככלומר נוצרה הודעה ממוגנת ספציפי (הלקוח) לשרת.

השרת מפיץ את ההודעה למשתמש ספציפי "Shira" הנבחר ונוצרה הודעה בחזרה גם.



כאשר גם נ עבור ל-Wireshark, נבחר פקטה של ההודעה.

נבחר את פקטה Frame 66 (פקטה של ההודעה) לצורך הדגמה ונוכל לראות את תוכן ההודעה בצורה טקסט וגם בקוד ASCII. כמובן נראה גם את שם המשתמש השולח.



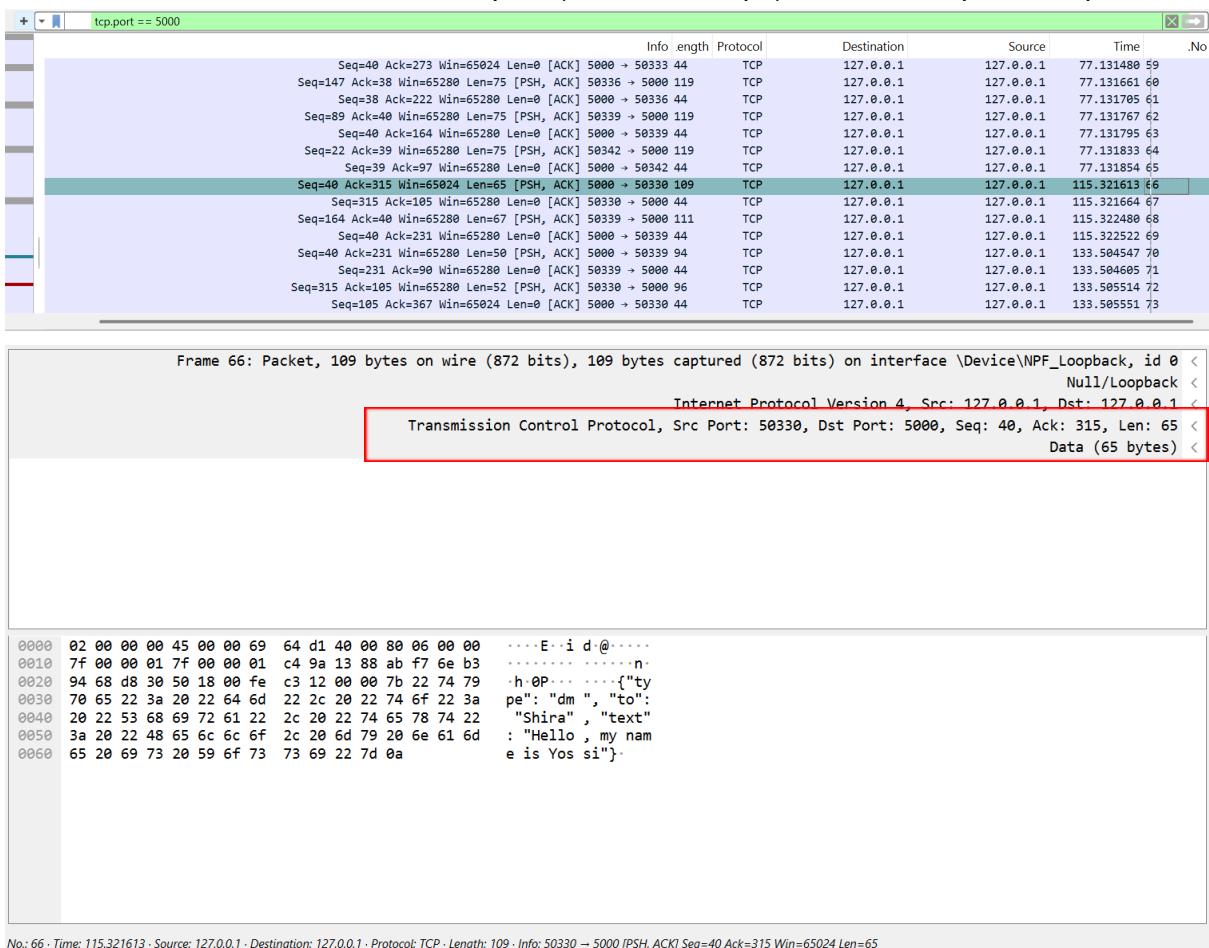
## ניתוח תעבורת של היחסן עד שכבת הרשות (כולל) (המשר)

### שכבת התעבורת:

ההודעה מועברת ב프וטוקול TCP.

כפי שŁמְדָנוּ בהרצאה, זה פרוטוקול אמין, וגם שומר על סדר ההודעות (חוֹזֶב במשק צ'אַט).

נבחר כמו קודם את פקטה של Frame 66 (פקטה של ההודעה) לצורך הדגמה.



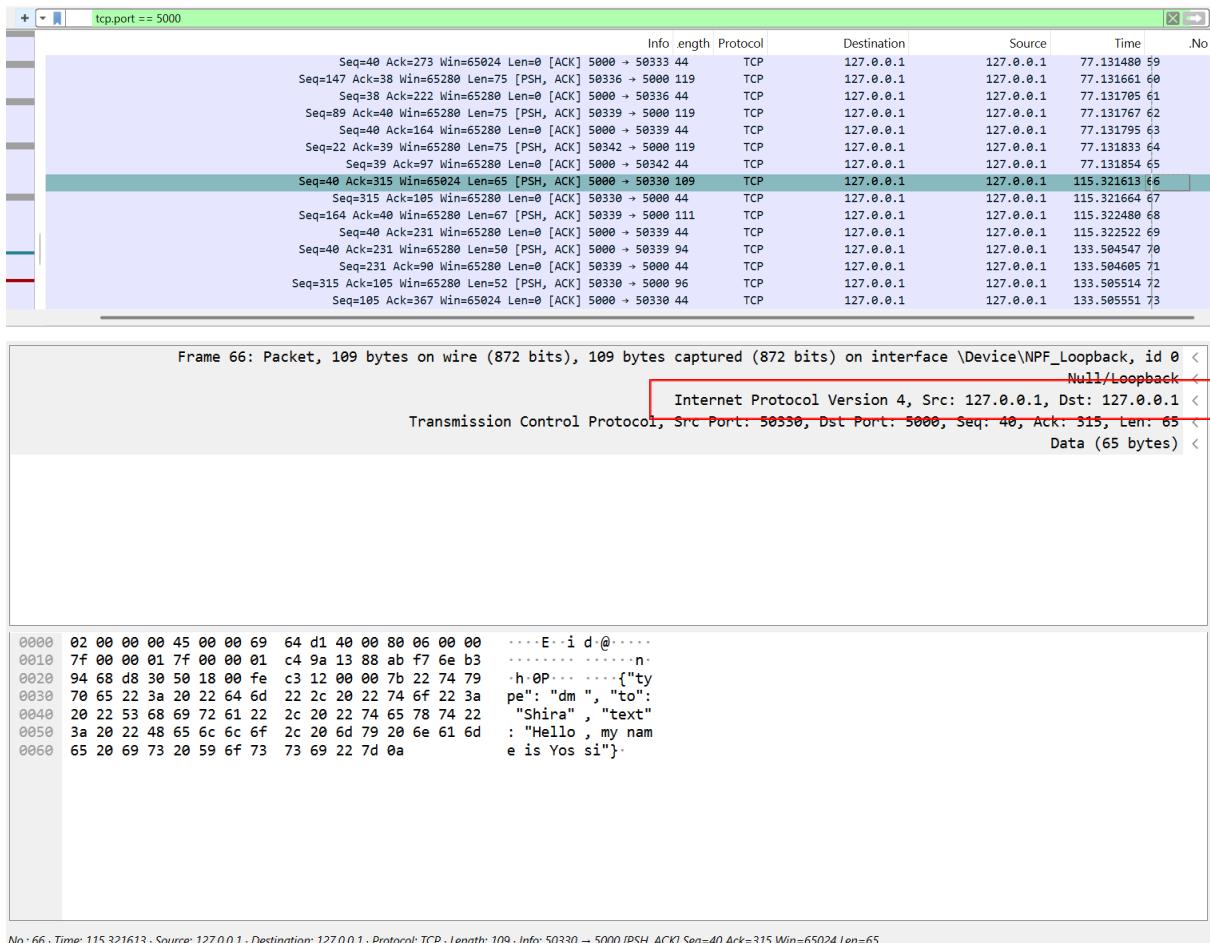
ניתן לראות שימוש בפרטיו, פקטה 66 (Transmission Control Protocol, TCP):

- ❖ שימוש בפרוטוקול TCP (Transmission Control Protocol)
- ❖ פורט מקורה: 50330, זה הפורט הזמן של המשתמש.
- ❖ פורט יעד: 5000, זה הפורט הקבוע של השרת.
- ❖ דגלי ACK ו-PSH: העברת נתונים.
- ❖ ACK-ACK - אישור על קבלת נתונים בצורה תקינה.
- ❖ אוֹרֵךְ הנתונים: byte 65

## ניתוח תעבורת של היחסן עד שכבת הרשת (כולל) (המשר)

שכבות הרשת:

בשכבה הרשת ניתן לראות באמצעות Wireshark את השימוש ב프וטוקול IPv4.



ניתן לראות ש坎坷 מופיע:

- ❖ כתובות מקור: 127.0.0.1
- ❖ כתובותיעד: 127.0.0.1

זה נכון שהכתובת המקור זהה לכתובת היעד משוםSCP התקשורת בוצעה על localhost. כמובן, כאן הפרויקט בוצע על מחשב יחיד, כל המשתמשים/לקוחות והשרות נמצאים על אותו המחשב.

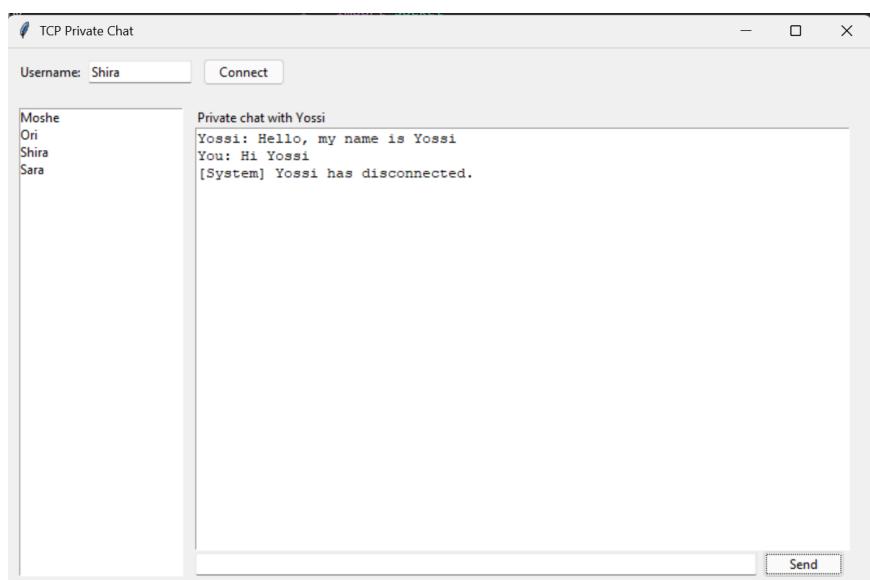
## ניתוח תעבורת של היבום עד שכבת הרשת (כולל) (המשר)

### ניתוח תעבורת של ניוטוק משתמש

המשתמש "Yossi" יצא מהמערכת בניתוק פטאומי מהמערכת.  
(ניתן גם לראות שאקן המערכת מטפלת במקרים של ניוטוק פטאומי של משתמש או מקרי קצה,, היא עדין מתירה וمعدכנת, וכמובן השירות ושאר המשתמשים ממשיכים לפעול תקין).  
cut ננתח את התעבורת של הניוטוק.

### שכבת היבום:

המשתמש "Yossi" יצא מהמערכת במהלך השיחה עם ". Shira".  
Shira קיבלה עדכון על כך בז'אט, המשתמש "Yossi" לא נמצא יותר בסרגל הצד.  
בנוסף השירות המרכזי עדכן שהמשתמש "Yossi" התנתק מהמערכת.



The screenshot shows a terminal window with the following log output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... X
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:50330
[SERVER] User 'Yossi' connected
[SERVER] New connection from 127.0.0.1:50333
[SERVER] User 'Moshe' connected
[SERVER] New connection from 127.0.0.1:50336
[SERVER] User 'Ori' connected
[SERVER] New connection from 127.0.0.1:50339
[SERVER] User 'Shira' connected
[SERVER] New connection from 127.0.0.1:50342
[SERVER] User 'Sara' connected
[SERVER] User 'Yossi' disconnected
```

To the right of the terminal is a sidebar showing a file browser with several Python files listed:

- python
- powershell
- python
- python
- python
- python

## ניתוח תעבורת של היחס עד שכבת הרשות (כולל) (המשר)

שכבת התעborות:

נفالטר את הפילטר הנכון על מנת למצוא את הפקטה המתאימה  $(\text{tcp.flags.reset} == 1)$ .  
נראה את הפקטה המתאימה, פקטה 74.

No.	Date	Time	Source	Destination	Protocol	Length	Info
74		167.079065	127.0.0.1	127.0.0.1	TCP	44	Seq=105 Ack=367 Win=0 Len=0 [RST, ACK] 5000 → 50330

Frame 74: Packet, 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF\_Loopback, id 0 < Null/Loopback < Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 < Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 105, Ack: 367, Len: 0 <

0000	02 00 00 00 45 00 00 28	64 d9 40 00 80 06 00 00	.....E..( d@....
0010	7f 00 00 01 7f 00 00 01	c4 9a 13 88 ab f7 6e f4	..... ....n.
0020	94 68 d8 64 50 14 00 00	51 f2 00 00	h.dP... Q...

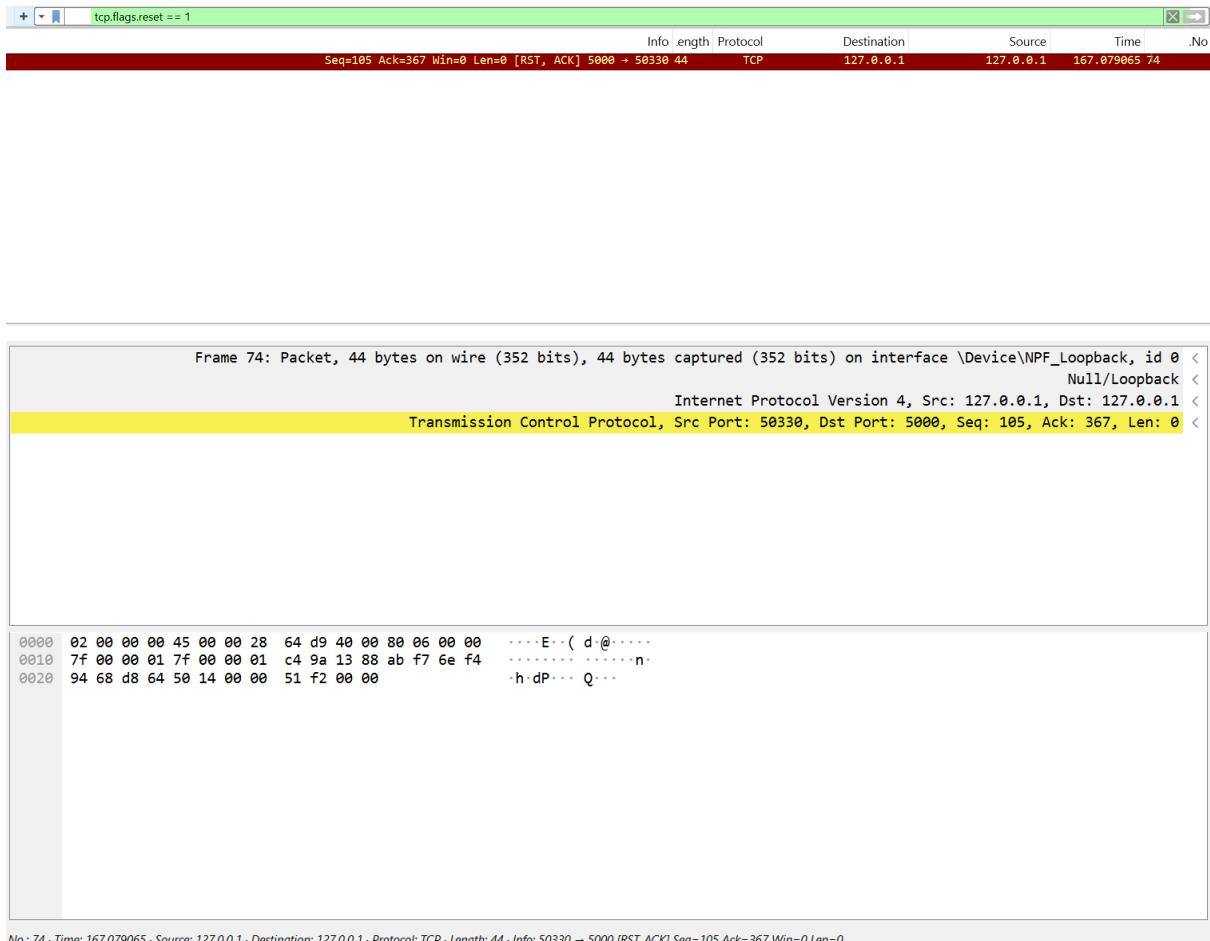
No.: 74 · Time: 167.079065 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 44 · Info: 50330 → 5000 [RST, ACK] Seq=105 Ack=367 Win=0 Len=0

- ❖ RST סגירה מיידית של החיבור בשל ניתוק פתאומי.
- ❖ ACK אישור קבלת נתונים.
- ❖ Len = 0 תיקין כי אין נתונים, זה לבקרה.
- ❖ פורט ממקור: 50330, זה ה포רט הזמן של המשתמש.
- ❖ פורט יעד: 5000, זה הפורט הקבוע של השרת

## ניתוח תעבורת של היחסם עד שכבת הרשת (כולל) (המשר)

שכבת הרשת:

גם כאן ניתן לראות כמו קודם בעזרת wireshark את השימוש בפרוטוקול IPv4.



ניתן לראות שאכן מופיע:

- ❖ כתובות מקור: 127.0.0.1
- ❖ כתובותיעד: 127.0.0.1

כפי שרשמה קודם,

זה נכון שהכתובת המקור זהה לכתובת היעד משום שככל התקשרות בוצעה על localhost. כמובן, כאן הפרויקט בוצע על מחשב יחיד, כל המשתמשים/локות והשרת נמצאים על אותו המחשב.

## סיכום ניתוח תעבורת של היחסם עד שכבת הרשת (כולל)

לסיכום, ניתן לראות על ידי ניתוח תעבורת ב Wireshark את המידע עבור דרך שכבת התעborת ועד שכבת הרשת.

נשלחה הודעה משתמש ספציפי למשתמש אחר ביצ'אט (או התקבל ניתוק משתמש ספציפי), ההודעה נארחת ב TCP בשכבת התעborת וועברת דרך IPv4 בשכבת הרשת.  
בנוסף, ניתן לומר שאכן המידע עבור דרך תקשורת אמינה ומוסדרת כפי שאמור להיות בפרוטוקול TCP.  
★ קובץ הלכידה בWireshark נשמר בפורמט pcap และ מצוי בפרויקט.

## תיאור שימוש במבנה מלאכותית

במהלך העבודה על הפרויקט נעשה שימוש במבנה מלאכותית (GPT) לצורך למידה והבנה של החומר בצורה מעמיקה וטובה יותר. בנוסף היה שימוש במבנה מלאכותית לצורך עזרה טכנית.

### מטרות שימוש

- ❖ שאלות מסוימות על החומר התיאורטי על מנת להבין אותו בצורה יותר מעמיקה ונכונה.
- ❖ עזרה טכנית בכתיבת קוד CSV בחלק הראשון של הפרויקט (כפי שרשום שניתן לעשויות בהוראות)
- ❖ עזרה טכנית בכתיבת הקוד (שרת-לקוח) בחלק השני של הפרויקט (לצער לי ידע עמוק בch0tPy ובכתיבת קוד שרת-לקוח, ולכן החלטתי לארח עזרה טכנית בWireshark כדי לוודא את עצמו).
- ❖ "סיעור מוחות" בנוגע לנימוח נתונים בWireshark כדי לוודא את עצמו.
- ❖ הסבר על חלקים מסוימים בקוד לצורך הבנת המרcura.

### דוגמאות לפירומפטים

- ❖ "תסביר לי כיצד עובד חיבור שרת-לקוח בפרוטוקול TCP בצורה יותר מעמיקה ומובנת"
- ❖ "tabuna li kovetz CSV lfi horotot haProjectu wlofi hdogma"
- ❖ "עוזרתך בהכנה יחד את קבצי קוד בch0tPy לצד שרת ולצד קוד Wireshark (אין לי ידע עמוק וניסיון בch0tPy בכתיבת קוד שרת-לקוח לצער)"
- ❖ "עכשו שס"ימנו להכין את קבצי הקוד, בוא נعبر חלק קצת בקוד ונבין בדיק מה הוא עשה ומה תפקידו"
- ❖ תודא שאכן הנתון שניתרתי מה Wireshark בנוגע לשכבות התעבורה בחלק זהה נכון, במידה ולא רק תסמן לי מה לא נכון שאבדוק את עצמו"