

# כתיבת יישום רשות וניהוח תעבורה

קורס: תקשורת מחשבים  
פרויקט גמר – ניתוח תעבורה בפרוטוקול IP/TCP

## 1. מבוא

בחילק זה של הפרויקט פותחה מערכת צ'אט מבוססת **Sockets בפרוטוקול TCP**, במטרה להבין לעומק את עיקרונות התקשרות ברשת, העבודה במודל שרת-לקוח וניהוח תעבורה בפועל באמצעות Wireshark. לאחר מימוש המערכת בוצעה לכידת תעבורה ונոתחו המנות עד **שכבת הרשת (כולל)**.

## 2. תיאור כללי של המערכת

המערכת פועלת במבנה **Client–Server**:

- **שרת (Server)**:  
השרת מאמין לחברים נכנים, מנהל משתמשים מחוברים, ומתחויר בין לקוחות לצורךפתיחה צ'אט פרטי בין שני משתמשים לפי שם.
- **לקוחות (Clients)**:  
מומשו שני סוגי לקוחות:
  - לקוח טקסטואלי (CLI)
  - לקוח עם ממושך גרפי (GUI) מבוסס Tkinter

התקשורת מתבצעת באמצעות הודעות JSON המועברות מעל TCP.

## 3. מבנה הקוד

קבצים עיקריים:

- `server.py` – מימוש השרת

• – ליקוח טקסטואלי `client_cli.py`

• – ליקוח עם ממשך גרפי `client_gui.py`

#### עקרונות מימוש:

• שימוש ב-`TCP (socket.SOCK_STREAM`

• תמיכה בריבוי ליקוחות באמצעות `thread` לכל ליקוח

• סנכרון באמצעות `lock`

• טיפול בנייטוקים ושגיאות תקשורת

## 4. פרוטוקול התקשרות (שכבה היעשות)

המערכת משתמש בפרוטוקול יישום פשוט בפורמט JSON:

#### הודעות מהלקוח לשרת:

• – התחברות למערכת `join {username}`

• – בקשה לפתיחת צ'אט פרטי `chat_request {to`

• – שליחת הודעה `chat {message}`

• – יציאה מצ'אט `leave_chat`

• – ניתוק מסודר `quit`

#### הודעות מהשרת ללקוח:

• – הודעות מערכת `system {message}`

• – הודעות שגיאה `error {message}`

– אישור פתיחת צ'אט {chat\_started {with •

– הודעת צ'אט כניסה {chat {from, message •

## 5. הוראות התקנה והרצה

### דרישות:

- Python 3.10 ומעלה

VS code •

לפתוח תיקייה SERVER\_CLIENT\_CHAT\_PART\_2

### הרצה:

1. הרצת השרת:

`python server.py`

2. הרצת לוח טקסטואלי:

`python client_cli.py`

3. הרצת לוח גרפי:

`python client_gui.py`

ניתן להריץ מספר לוחות במקביל ולפתח צ'אטים פרטיים ביניהם.

## 6. דוגמאות קלט ופלט

קלט (ליקוח):

```
/chat bob  
hi
```

פלט (ליקוח):

```
[chat started with bob  
bob: hi
```

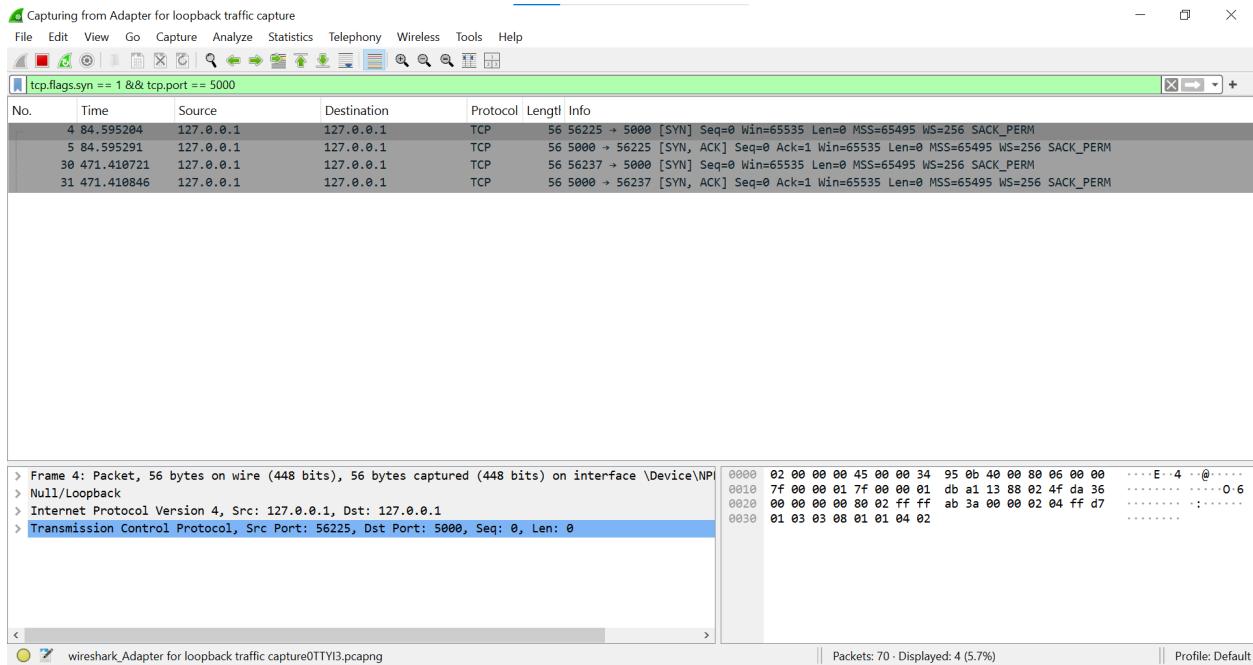
## 7. ניתוח תעבורת ב-Wireshark

### 7.1 הקמת חיבור (TCP Handshake)

בוצעה לכידת תעבורת בזמן חיבור לקו לשרת.

פילטר:

```
tcp.flags.syn == 1 && tcp.port == 5000
```



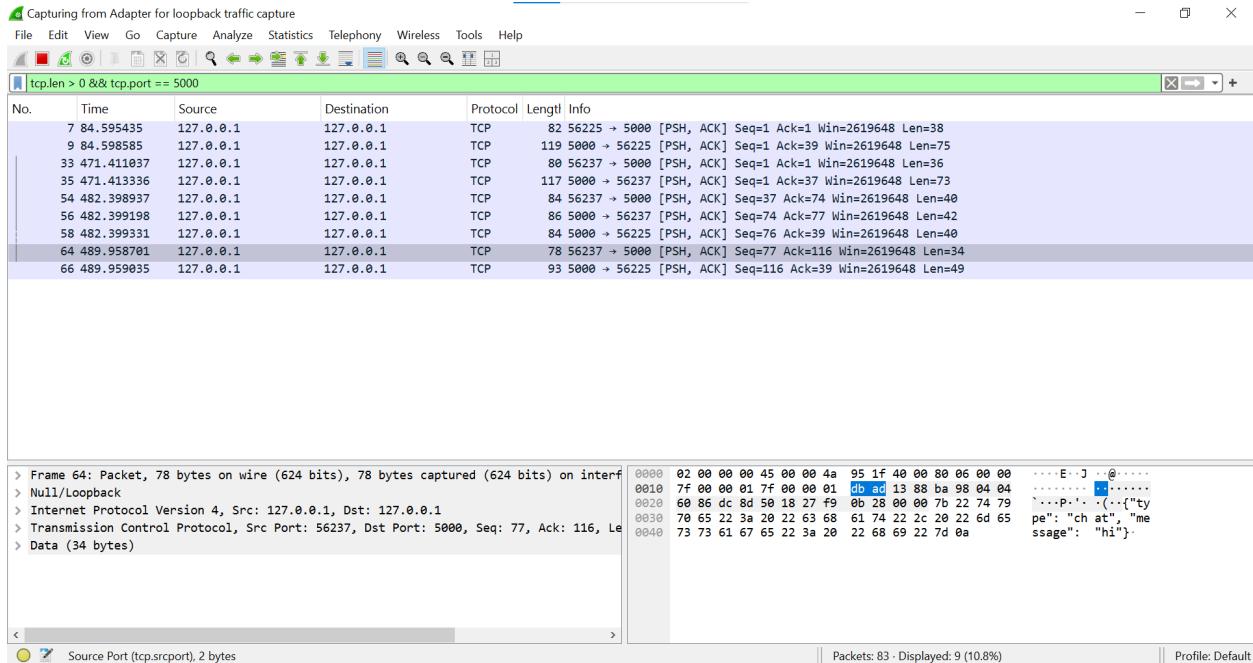
בצלום ניתן לראות את שלב הקמת החיבור ב프וטוקול TCP (3-Way Handshake), הכולל שליחת SYN מהלך וקבלת SYN/ACK מהשרת לפני תחילת העברת הנתונים.

## 7.2 העברת הודעות (Data)

לאחר הקמת החיבור נשלחו הודעות צ'אט בין הלקוחות.

**פילטר:**

```
tcp.len > 0 && tcp.port == 5000
```



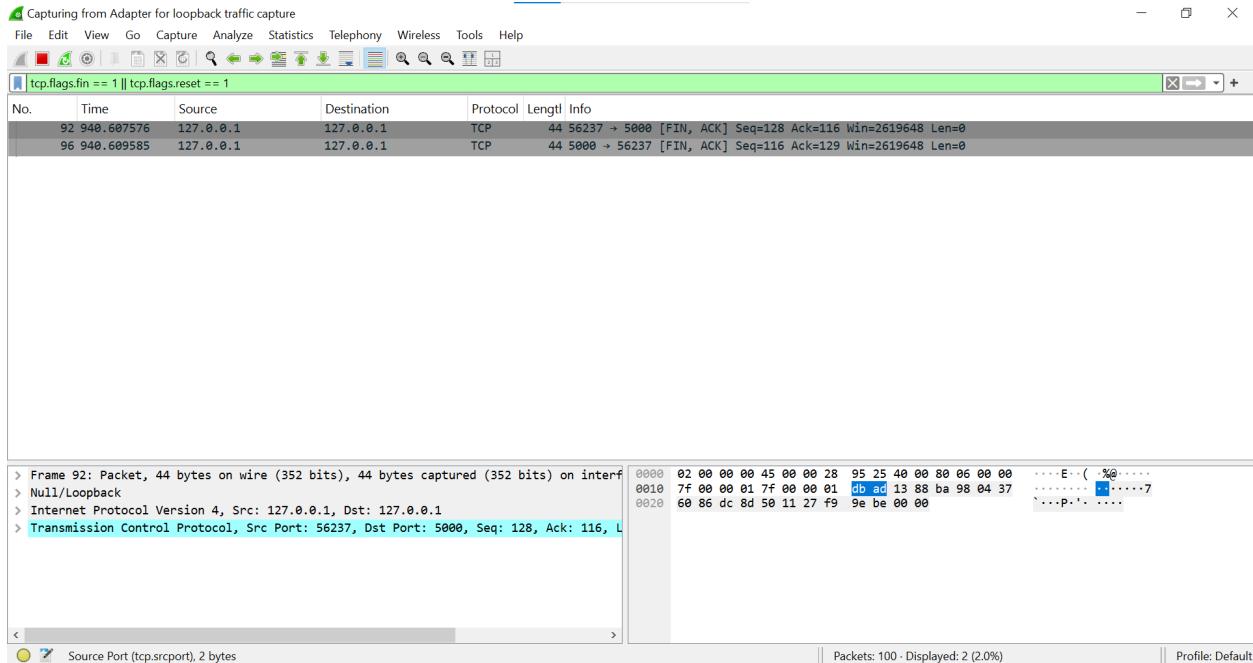
בצילום נראות מנוגת TCP עם הדגלים PSH ו-ACK, המכילות Payload של שכבת היפר טקסט בפורמט JSON, לדוגמה הודעת צ'אט שנשלחה בין משתמשים.

### 7.3 סיום חיבור TCP

בסיום השיחה או ביציאה מהמערכת נספתחה סגירת החיבור.

**פילטר:**

```
tcp.flags.fin == 1 || tcp.flags.reset == 1
```



הצילום מציג סיום תקשורת TCP באמצעות מנוט FIN ו-ACK, המעידות על ניתוק מסודר בין הלקוח לשרת.

ניתן לראות גם בקובץ `chat_part2.pcapng` בתוך התיקייה.

## 8. שימוש בבינה מלאכותית

במהלך הפרויקט נעשה שימוש בבינה מלאכותית לצורכי:

- **תכנון ארכיטקטורת המערכת**
- **בדיקה תקינות לוגית של הפרטוקול**

- שיפור תיעוד ובהירות הקוד

**מטרת השימוש:** סיוע בתהיליך הפיתוח ללא שימוש בקוד מוכן.

## 9. סיכום

בפרויקט זה פותחה מערכת צ'אט מלאה מבוססת TCP, הכוללת שרת ולקוחות, תמיינה בריבוי משתמשים וניתוח תယborות רשות בפועל.  
הפרויקט הדגים יישום מעשי של עקרונות תקשורת מחשבים וניתוח פרוטוקולים באמצעות Wireshark.