

# כתיבת יישום רשת וניתוח תעבורה

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

## 1. מבוא

בחלק זה של הפרויקט פותחה מערכת צ'אט מבוססת **Sockets בפרוטוקול TCP**, במטרה להבין לעומק את עקרונות התקשורת ברשת, עבודה במודל שרת-לקוח וניתוח תעבורה בפועל באמצעות Wireshark. לאחר מימוש המערכת בוצעה לכידת תעבורה וניתוח המנות עד **שכבת הרשת (כולל)**.

## 2. תיאור כללי של המערכת

המערכת פועלת במבנה **Client-Server**:

- **שרת (Server)**  
השרת מאזין לחיבורים נכנסים, מנהל משתמשים מחוברים, ומתווך בין לקוחות לצורך פתיחת צ'אט פרטי בין שני משתמשים לפי שם.

- **לקוחות (Clients)**  
מומשו שני סוגי לקוחות:

- לקוח טקסטואלי (CLI)

- לקוח עם ממשק גרפי (GUI) מבוסס Tkinter

התקשורת מתבצעת באמצעות הודעות JSON המועברות מעל TCP.

## 3. מבנה הקוד

**קבצים עיקריים:**

- `server.py` – מימוש השרת

- `client_cli.py` – לקוח טקסטואלי
- `client_gui.py` – לקוח עם ממשק גרפי

### עקרונות מימוש:

- שימוש ב-`socket.SOCK_STREAM` (TCP)
- תמיכה בריבוי לקוחות באמצעות **thread לכל לקוח**
- סנכרון באמצעות `lock`
- טיפול בניתוקים ושגיאות תקשורת

## 4. פרוטוקול התקשורת (שכבת היישום)

המערכת משתמשת בפרוטוקול יישום פשוט בפורמט JSON:

### הודעות מהלקוח לשרת:

- `{join {username` – התחברות למערכת
- `{chat_request {to` – בקשה לפתיחת צ'אט פרטי
- `{chat {message` – שליחת הודעה
- `leave_chat` – יציאה מצ'אט
- `quit` – ניתוק מסודר

### הודעות מהשרת ללקוח:

- `{system {message` – הודעות מערכת
- `{error {message` – הודעות שגיאה

- `{chat_started {with` – אישור פתיחת צ'אט
- `{chat {from, message` – הודעת צ'אט נכנסת

## 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 לקוח טקסטואלי:

```
python client_cli.py
```

3. הרצת לקוח גרפי:

```
python client_gui.py
```

ניתן להריץ מספר לקוחות במקביל ולפתוח צ'אטים פרטיים ביניהם.

## 6. דוגמאות קלט ופלט

קלט (לקוח):

```
/chat bob  
hi
```

פלט (לקוח):

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

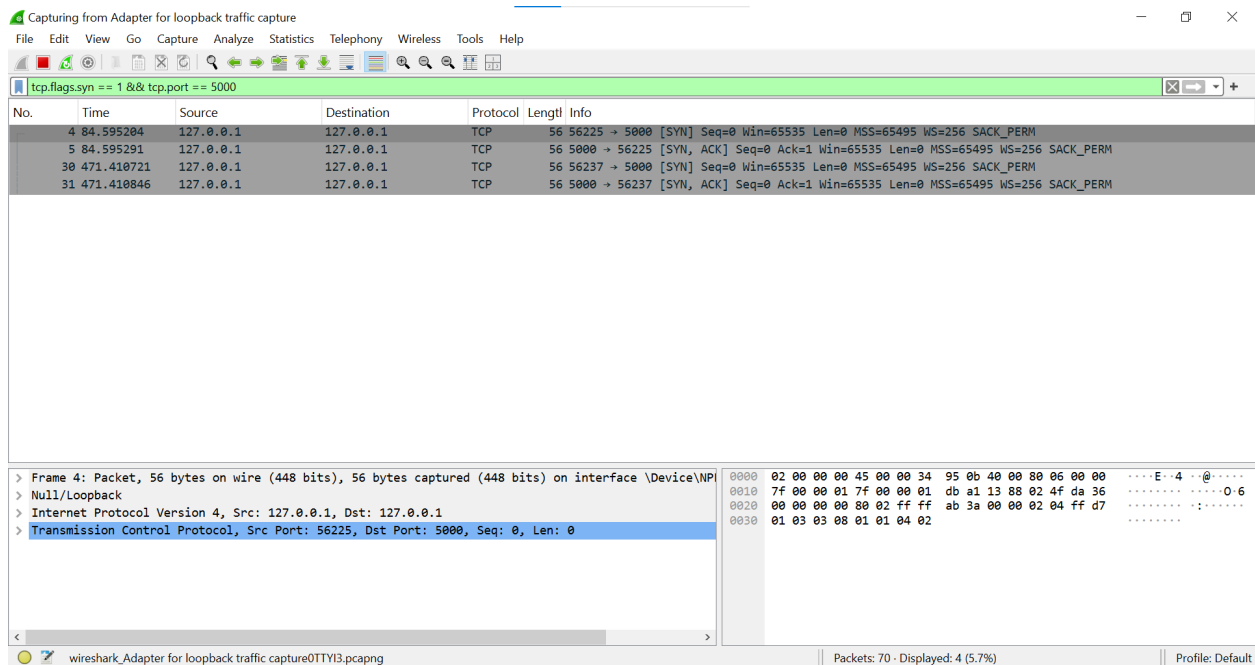
## 7. ניתוח תעבורה ב-Wireshark

### 7.1 הקמת חיבור TCP (Handshake)

בוצעה לכידת תעבורה בזמן חיבור לקוח לשרת.

פילטר:

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



בצילום ניתן לראות את שלב הקמת החיבור בפרוטוקול TCP (3-Way Handshake), הכולל שליחת SYN מהלקוח 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`

Capturing from Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
7	84.595435	127.0.0.1	127.0.0.1	TCP	82	56225 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=38
9	84.598585	127.0.0.1	127.0.0.1	TCP	119	5000 → 56225 [PSH, ACK] Seq=1 Ack=39 Win=2619648 Len=75
33	471.411037	127.0.0.1	127.0.0.1	TCP	80	56237 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=36
35	471.413336	127.0.0.1	127.0.0.1	TCP	117	5000 → 56237 [PSH, ACK] Seq=1 Ack=37 Win=2619648 Len=73
54	482.398937	127.0.0.1	127.0.0.1	TCP	84	56237 → 5000 [PSH, ACK] Seq=37 Ack=74 Win=2619648 Len=40
56	482.399198	127.0.0.1	127.0.0.1	TCP	86	5000 → 56237 [PSH, ACK] Seq=74 Ack=77 Win=2619648 Len=42
58	482.399331	127.0.0.1	127.0.0.1	TCP	84	5000 → 56225 [PSH, ACK] Seq=76 Ack=39 Win=2619648 Len=40
64	489.958701	127.0.0.1	127.0.0.1	TCP	78	56237 → 5000 [PSH, ACK] Seq=77 Ack=116 Win=2619648 Len=34
66	489.959035	127.0.0.1	127.0.0.1	TCP	93	5000 → 56225 [PSH, ACK] Seq=116 Ack=39 Win=2619648 Len=49

> Frame 64: Packet, 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 56237, Dst Port: 5000, Seq: 77, Ack: 116, Len: 34

> Data (34 bytes)

```

0000 02 00 00 00 45 00 00 0a 95 1f 40 00 80 06 00 00  ....E...@.....
0010 7f 00 00 01 7f 00 00 01 db ad 13 88 ba 98 04 04  ....P.....{ty
0020 60 86 dc 8d 50 18 27 f9 0b 28 00 00 7b 22 74 79  ....P.....{ty
0030 70 65 22 3a 20 22 63 68 61 74 22 2c 20 22 6d 65  pe": "ch at", "me
0040 73 73 61 67 65 22 3a 20 22 68 69 22 7d 0a      ssage": "hi"}-

```

Source Port (tcp.srcport), 2 bytes

Packets: 83 - Displayed: 9 (10.8%)

Profile: Default

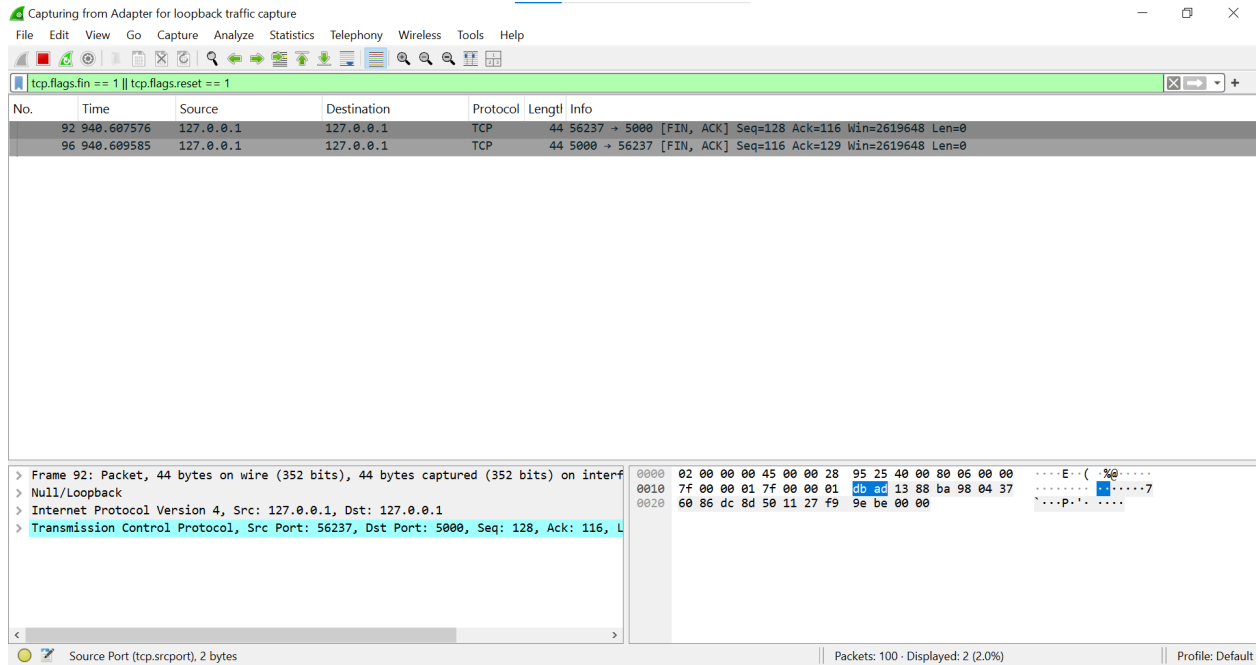
בצילום נראות מנות TCP עם הדגלים PSH ו-ACK, המכילות Payload של שכבת היישום בפורמט JSON, לדוגמה הודעת צ'אט שנשלחה בין משתמשים.

## 7.3 סיום חיבור TCP

בסיום השיחה או ביציאה מהמערכת נצפתה סגירת החיבור.

פילטר:

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



הצילום מציג סיום תקשורת TCP באמצעות מנות 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 לצורך:

- תכנון ארכיטקטורת המערכת
- בדיקת תקינות לוגית של הפרוטוקול

- שיפור תיעוד ובהירות הקוד

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

## 9. סיכום

בפרויקט זה פותחה מערכת צ'אט מלאה מבוססת TCP, הכוללת שרת ולקוחות, תמיכה בריבוי משתמשים וניתוח תעבורת רשת בפועל.

הפרויקט הדגים יישום מעשי של עקרונות תקשורת מחשבים וניתוח פרוטוקולים באמצעות Wireshark.