

# דוח מסכם - פרויקט ברשתות תקשורת מחשבים

## מגשים: אלון דולב ושירן קנדוב

### חלק 1 - אריזת נתונים ולכידת מנות

#### 1. יצירת קובץ ה-CSV

לצורך הפרויקט יצרנו קובץ CSV שמכיל הודעות בשכבת היישום (Application Layer).  
בחרנו להשתמש בפרוטוקול HTTP משום שהוא פשוט, ברור וקל לזיהוי ב-Wireshark.

שם הקובץ:

group01\_http\_input.csv

העמודות בקובץ:

msg\_id,app\_protocol,src\_app,src\_port,dst\_app,dst\_port,message,timestamp

הקובץ כולל סדרת בקשות ותגובות HTTP כמו GET, POST, 200 OK ו-404 Not Found, כדי לדמות תקשורת בין דפדפן לשרת.

### 2. הרצת המחברת Jupyter והדמיית אריזת הנתונים

הפעלנו את המחברת שסופקה בקורס מקומית דרך VSCode. זה מאפשר ליצור חבילות TCP אמיתיות שנשלחות דרך מערכת ההפעלה.

במהלך העבודה בוצעו השלבים הבאים:

- טעינת קובץ ה-CSV באמצעות pandas.

- בדיקת תקינות מבנה ה-CSV.

- בניית כותרות של שכבות TCP/IP:

כותרת IP (גרסה, כתובות מקור ויעד, אורך חבילה, checksum).

כותרת TCP (src\_port, dst\_port, sequence, flags).

- אריזת ההודעה: IP -> TCP -> Application.

- שליחת החבילות בפועל לכתובת 127.0.0.1 באמצעות Raw sockets או Scapy בהתאם למערכת.

המחברת שלחה בהצלחה את כל ההודעות מה-CSV כחבילות TCP ל-loopback.

```
jupyter_networks.ipynb X
jupyter_networks.ipynb > M4 TCP/IP Encapsulation Project – Student Guide (English) > M4 Step 6 — Analyze and Explain
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ... Python 3.12

# find interface name for windows
if IS_WINDOWS and HAVE_SCAPY:
    try:
        print('\n'.join(get_if_list()))
    except Exception as e:
        print('could not list interfaces:', e)

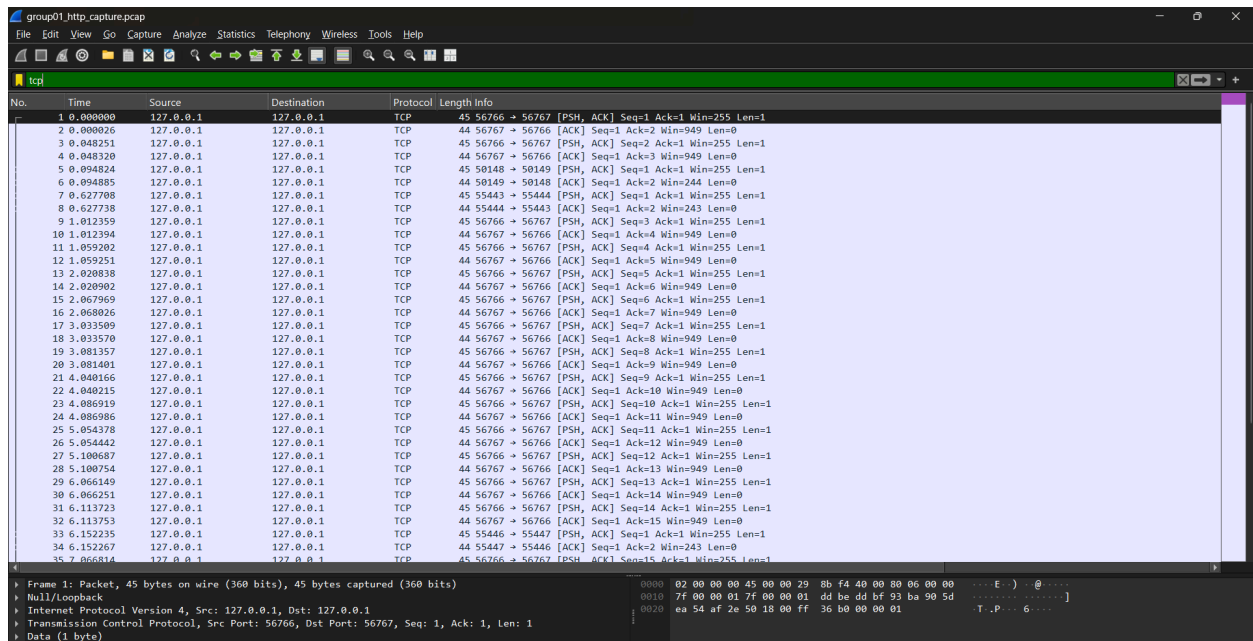
[11] ✓ 0.0s Python

...
\\Device\\NPF_{F02D51C8-917D-4349-8FA4-39D6C8ECF653}
\\Device\\NPF_{D29AA0A8-1E7C-499A-8205-685F0DDE9324}
\\Device\\NPF_{21542F33-6496-4919-9954-73A777D39B9B}
\\Device\\NPF_{A515C58C-099B-4266-B177-1125EEC1AFD6}
\\Device\\NPF_{C506BFB2-DC1B-4262-8D35-F29F6099D0C4}
\\Device\\NPF_{9E21BAA4-BF91-400F-B98D-CB5634F779B4}
\\Device\\NPF_{C339A1A6-3559-46F1-8C9A-9A1D43CEC3EE}
\\Device\\NPF_{E1B195A0-4E92-4105-BE93-44F739B00D45}
\\Device\\NPF_{Loopback}

# 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) + payload
hexdump(pkt_preview)

[17] ✓ 0.0s Python

...
0000 45 00 00 3e 32 59 00 00 40 06 4a 5f 7f 00 00 01  E...2Y...@.J....
0010 7f 00 00 01 69 a6 30 39 f1 67 be 58 00 00 00 00  ....i.09.g.X....
0020 50 02 ff ff 54 57 00 00 48 65 6c 6c 6f 20 50 61  P...TW..Hello Pa
0030 63 6b 65 74 20 28 70 72 65 76 69 65 77 29      cket (preview)
```



### 3. לכידת התעבורה ב-Wireshark

הליכידה בוצעה באמצעות Wireshark על ממשק Adapter for loopback traffic capture, מאחר והתקשורת בוצעה לכתובת 127.0.0.1. בוצע סינון תצוגה באמצעות הפילטר: tcp.

בסקרינשוט ניתן לראות סדרת חבילות TCP שנשלחו בין כתובת המקור והיעד 127.0.0.1, כאשר כל חבילה מייצגת הודעה שנשלחה מקובץ ה-CSV לאחר תהליך האריזה (Encapsulation).

בדוגמה המוצגת נבחרה חבילת TCP הכוללת Payload באורך 1 byte, המתאימה להודעת Application-מה-CSV.

בכותרת ה-TCP ניתן לראות:

פורט מקור ופורט יעד תואמים לערכים שהוגדרו בקובץ ה-CSV, דגלים מסוג PSH ו-Ack המעידים על שליחת נתוני אפליקציה, ומספרי Sequence ו-Acknowledgment המשקפים את מנגנון בקרת הזרימה של TCP.

בכותרת ה-IP מופיעות כתובות המקור והיעד 127.0.0.1, דבר המאשר שהשליחה בוצעה דרך ממשק loopback. ב-Payload של החבילה מופיעים נתוני ההודעה משכבת ה-Application, כפי שהוגדרו בשורת ה-CSV המתאימה.

לכידה זו מדגימה את תהליך האריזה המלא:

הודעת Application, עטיפה ב-TCP, עטיפה ב-IP, שליחה כחבילת רשת שנקלטת ב-Wireshark.

ניתן לקשר בין שורות קובץ ה-CSV לבין החבילות שנצפו ב-Wireshark באמצעות שדה msg\_id. לדוגמה, שורה עם msg\_id=2 בקובץ ה-CSV, המכילה בקשת GET, מופיעה ב-Wireshark כחבילת TCP עם Payload תואם.

התאמה זו מאשרת שכל הודעת Application-מה-CSV נאזרה, נשלחה ונלכדה כמתואר. לאחר הליכידה שמרתי את הקובץ כ: group01\_http\_capture.pcap.

## חלק 2 – מימוש מערכת צ'אט מבוססת TCP וניתוח תעבורה

### 1. מבנה כללי של המערכת

במסגרת חלק זה מימשנו מערכת צ'אט מבוססת TCP הכוללת שרת (server.py) ושני לקוחות או יותר עם ממשק גרפי (client\_gui.py). השרת מאזין לחיבורים נכנסים, שומר כל לקוח בשם ייחודי, ומשמש כמתווך בין לקוחות המעוניינים לשוחח זה עם זה. כל התקשורת מבוצעת מעל TCP תוך שימוש ב-sockets בלבד, ללא שימוש בספריות מוכנות לניהול שרתים או לקוחות.

### 2. השרת – server.py

השרת יוצר socket מוג TCP, מאזין בפורט 5000 ומטפל במספר לקוחות במקביל באמצעות threads. בעת התחברות, כל לקוח שולח לשרת את שמו הייחודי. השרת תומך בפקודות טקסטואליות פשוטות כגון MSG, CHAT, LIST, HELLO ו-QUIT, ומנהל מיפוי בין לקוחות מחוברים. כאשר לקוח מבקש לפתוח צ'אט עם לקוח אחר, השרת בודק אם הלקוח קיים ומתווך את ההודעות ביניהם. כל הודעה מועברת דרך השרת ולא ישירות בין הלקוחות. השרת כולל טיפול בניתוקים לא צפויים של לקוחות.

### 3. הלקוח – client\_gui.py

הלקוח יוצר חיבור TCP לשרת בפורט 5000 ומספק ממשק גרפי מבוסס Tkinter. הממשק מאפשר הזנת שם משתמש, התחברות לשרת, צפייה ברשימת משתמשים מחוברים, בחירת משתמש לצ'אט ושליחת הודעות טקסט בזמן אמת. קליטת הודעות מתבצעת באמצעות thread ייעודי, בעוד שה-thread הראשי אחראי על הממשק הגרפי ושליחת הודעות.

### 4. אופן ההרצה

1. מפעילים את השרת: python server.py
2. מפעילים מספר לקוחות: python client\_gui.py
3. כל לקוח מזין שם ייחודי ומתחבר לשרת.
4. הלקוח בוחר משתמש אחר מרשימת המשתמשים ופותח צ'אט.
5. לאחר החיבור, מתבצעת שיחה דו-כיוונית בזמן אמת.

המערכת תומכת במספר לקוחות בו-זמנית.

## 5. דוגמאות קלט ופלט

להלן דוגמאות לתקשורת טקסטואלית בין לקוח לשרת בפרוטוקול הצ'אט מעל TCP. הדוגמאות מייצגות את ההודעות שנשלחות בפועל דרך חיבור ה-socket, ללא תלות בממשק הגרפי.

תרחיש 1 – התחברות ורשימת משתמשים

```
Client → HELLO Alon
Server → SYSTEM Hello Alon, you are connected.
Client → LIST
Server → USERS Alon, Shiran
```

תרחיש 2 – פתיחת שיחה ושליחת הודעה

```
Client → CHAT Shiran
Server → SYSTEM You are now chatting with Shiran
Client → MSG Hello, how are you?
Server → CHAT Alon: Hello, how are you?
```

תרחיש 3 – ניתוק

```
Client → QUIT
Server → SYSTEM Disconnected
```

הודעות אלו מועברות על גבי חיבור TCP, מקודדות כמחרוזות טקסט, ומנותחות בצד השרת לצורך ניהול חיבורים, ניתוב שיחות והפצת הודעות בין משתמשים.

## 6. לכידת תעבורה ב-Wireshark

לכידת התעבורה בוצעה באמצעות Wireshark על ממשק Adapter for loopback traffic capture, מאחר והתקשורת בוצעה לכתובת 127.0.0.1.

שלבי הלכידה:

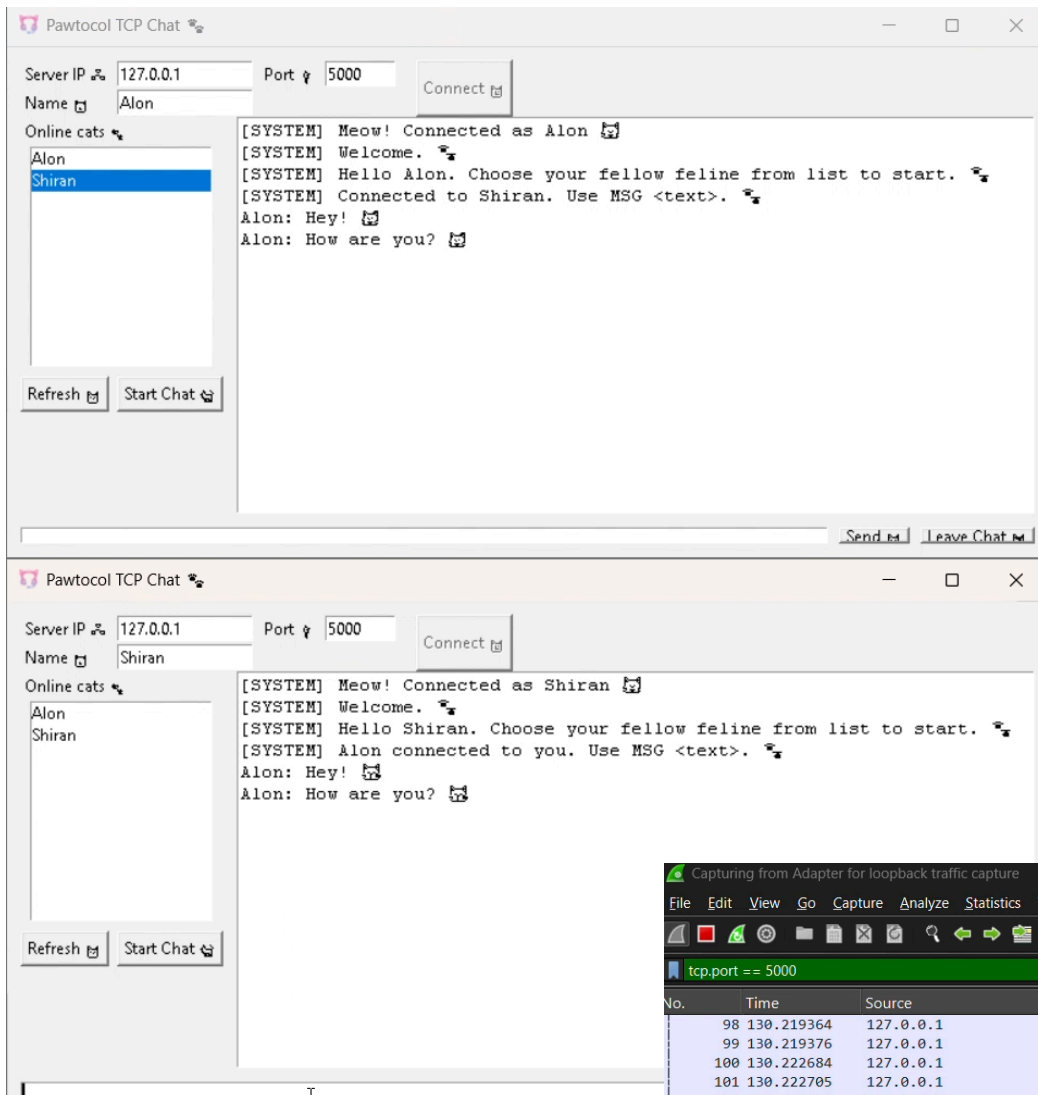
1. פתיחת Wireshark.
2. בחירת Adapter for loopback traffic capture.
3. שימוש בפילטר: tcp.port == 5000
4. התחלת Capture.
5. הפעלת השרת ושני לקוחות והרצת צ'אט.
6. עצירת הלכידה ושמירה לקובץ: group01\_chat\_capture.pcap

## 7. ניתוח תעבורה עד שכבת הרשת (IP + TCP)

ב-Wireshark ניתן לראות בבירור:

- פתיחת חיבור באמצעות 3-way handshake (TCP SYN, SYN-ACK, ACK).
- שימוש בפורט 5000 בצד השרת ובפורטים זמניים בצד הלקוחות.
- חבילות PSH ו-ACK בעת שליחת הודעות טקסט בין הלקוחות.
- ב-TCP Stream ניתן לראות את פקודות הפרוטוקול וההודעות שהוחלפו (HELLO, CHAT, MSG).
- בשכבת ה-IP נראות כתובות המקור והיעד 127.0.0.1.
- בשכבת ה-TCP נראים מספרי רצף, חלונות ו-checksum.

התעבורה יציבה, דו-כיוונית ועומדת בתקן TCP.



tcp.port == 5000

| No. | Time       | Source    | Destination | Protocol | Length | Info                           |
|-----|------------|-----------|-------------|----------|--------|--------------------------------|
| 98  | 130.219364 | 127.0.0.1 | 127.0.0.1   | TCP      | 62     | 5000 → 63900 [PSH, ACK] Seq=14 |
| 99  | 130.219376 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63900 → 5000 [ACK] Seq=14      |
| 100 | 130.222684 | 127.0.0.1 | 127.0.0.1   | TCP      | 49     | 63900 → 5000 [PSH, ACK] Seq=14 |
| 101 | 130.222705 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 5000 → 63900 [ACK] Seq=10      |
| 102 | 130.222802 | 127.0.0.1 | 127.0.0.1   | TCP      | 62     | 5000 → 63900 [PSH, ACK] Seq=10 |
| 103 | 130.222837 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63900 → 5000 [ACK] Seq=19      |
| 106 | 136.817727 | 127.0.0.1 | 127.0.0.1   | TCP      | 56     | 63899 → 5000 [PSH, ACK] Seq=12 |
| 107 | 136.817781 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 5000 → 63899 [ACK] Seq=12      |
| 108 | 136.817965 | 127.0.0.1 | 127.0.0.1   | TCP      | 88     | 5000 → 63899 [PSH, ACK] Seq=12 |
| 109 | 136.817995 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63899 → 5000 [ACK] Seq=29      |
| 110 | 136.818011 | 127.0.0.1 | 127.0.0.1   | TCP      | 90     | 5000 → 63900 [PSH, ACK] Seq=19 |
| 111 | 136.818027 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63900 → 5000 [ACK] Seq=19      |
| 116 | 144.665121 | 127.0.0.1 | 127.0.0.1   | TCP      | 53     | 63899 → 5000 [PSH, ACK] Seq=16 |
| 117 | 144.665145 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 5000 → 63899 [ACK] Seq=16      |
| 118 | 144.665297 | 127.0.0.1 | 127.0.0.1   | TCP      | 60     | 5000 → 63900 [PSH, ACK] Seq=19 |
| 119 | 144.665319 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63900 → 5000 [ACK] Seq=19      |
| 125 | 149.290533 | 127.0.0.1 | 127.0.0.1   | TCP      | 61     | 63899 → 5000 [PSH, ACK] Seq=16 |
| 126 | 149.290562 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 5000 → 63899 [ACK] Seq=16      |
| 127 | 149.290701 | 127.0.0.1 | 127.0.0.1   | TCP      | 68     | 5000 → 63900 [PSH, ACK] Seq=19 |
| 128 | 149.290729 | 127.0.0.1 | 127.0.0.1   | TCP      | 44     | 63900 → 5000 [ACK] Seq=19      |

## 8. שימוש בכלי AI - מטרות ודוגמאות פרומפטים

במהלך פיתוח הפרויקט נעשה שימוש בכלי בינה מלאכותית (AI) לצורך סיוע בתכנון, מימוש ותיעוד המערכת. השימוש ב-AI לא היה חלק מהריצה של המערכת עצמה אלא ככלי עזר למפתח.

מטרות השימוש ב-AI:

- הבנת דרישות הפרויקט ופירוקן למשימות פיתוח
- תכנון פרוטוקול תקשורת פשוט מעל TCP
- שיפור מבנה הקוד והטיפול במקרי קצה
- ניסוח והרחבת תיעוד טכני לדוח

דוגמאות לפרומפטים שנעשה בהם שימוש:

- *"Design a simple text-based protocol for a TCP chat server with multiple clients"*  
התקבל מבנה פקודות בסיסי (HELLO, LIST, CHAT, MSG, QUIT)
- *"How should a TCP server handle unexpected client disconnections in Python"*  
סיוע בהבנת טיפול ב-exceptions וניקוי חיבורים
- *"Explain TCP packet flow for a simple client-server chat application"*  
סיוע בכתובת סעיף ניתוח התעבורה בדוח
- *"Improve the clarity of this technical explanation for a networking course report"*  
שיפור ניסוחים בדוח המסכם

השימוש ב-AI תרם ליעילות הפיתוח ולדיוק התיעוד, אך כל הקוד נכתב ונבדק על ידינו.