

# דוח מסכם: פרויקט תקשורת מחשבים - חלק 1

מגישות: נתע גולזאיד 00 213487200, עדן זרביאן 212055024

[https://github.com/edenzarbian/networking\\_project](https://github.com/edenzarbian/networking_project)

## פרוטוקול HTTP:

פרוטוקול HTTP הוא פרוטוקול תקשורת המשמש להעברת מידע בשכבה האפליקצייתית (Application Layer) בין קlient (Client) לבין Server (Server).

- **הלקוח (Client)** - מיציג יישום קצה, כגון דפדפן או סקריפט (לדוגמה: קובץ Python), אשר שולח בקשה לשרת.
- **השרת (Server)** - מאזין לבקשת, מעבד אותה ומחזיר תשובה הכוללת קוד סטטוס (לדוגמה Not Found) וכן את המידע המבוקש.

הפרוטוקול פועל במודל בקשה תגובה (Request Response), שבו הלקוח יוזם את הבקשת והשרת מחזיר תגובה מתאימה.  
פרויקט זה נעשה שימוש ב프וטוקול HTTP לצורך הדגמת תעבורת רשות פשוטה וברורה, שנitinן לנתח בקלות באמצעות הכליל Wireshark.

HTTP פועל מעל פרוטוקול TCP בשכבה התעבורה, אשר אחראי על ייצור חיבור אמין בין הלקוח לשרת, על שמירת סדר הנתונים ועל שלמותם.

## :קובץ CSV:

חלק מהפרויקט יצרנו קובץ נתונים בפורמט CSV בשם group212055024\_fixed CSV, אשר משמש כקלט לייצור תעבורת הרשות.

הקובץ נוצר באופן ידני, במטרה לדמות הודעות בשכבה האפליקצייתית בצורה מבוקרת וברורה.

קובץ ה- CSV מיצג הודעות HTTP בשכבה האפליקצייתית לפני תחילת הארץיה לשכבות הנמוכות יותר של מודול TCP/IP. כל שורה בקובץ מדמה הודעה HTTP אחת הנשלחת מהלקוח אל השרת.

מבנה הקובץ כולל את השדות הבאים: msg\_id, app\_protocol, src\_app, dst\_app, message timestamp  
יצירה ידנית של הקובץ מאפשרת לנו שליטה מלאה בתוכן ההודעות, בסדר שליחתן ובזמן השיליחה, והקלת על ניתוח התעבורה שנוצרה בהמשך באמצעות Wireshark.

## תהליך ה- Encapsulation והרכבת המחברת Jupyter:

בשלב זה השתמשנו במחברת `raw_ipynb`, על מנת לדמות את התהליך אריזת הנתונים (Encapsulation) כפי שהוא מתבצע במודל IP/TCP.

הקוד במחברת אינו שולח הודעה רשות אמיטיות, אלא מדמה את התהליך הלוגי של העברת הודעה בין שכבות הרשת, בדומה לאופן שבו מערכת הפעלה מטפלת נתונים בעת שליחתם.

תהליך ה- Encapsulation מתבצע באמצעות הוספת כותרות (Headers) בכל שכבה. כותרות אלו מכילות מידע טכני הנדרש לצורכי ניתוב, זיהוי וטיפול נכון בהודעה על ידי המחשבים ברשת.

תהליך זה מדמה את אופן פעולה ה- Client היוצר את התעבורה, תעבורה שאotta ניתן לראות ולנתה לאחר מכן באמצעות הכליל Wireshark.

### פירוט תהליך האריזה לפי שכבות:

**שכבה היישום (Application Layer)** - בשלב זה הקוד קורא את שדה `message` מתוך קובץ CSV - Hello Packet או כהודעת HTTP פשוטה. ומשתמש בו כבסיס להודעה. לצורך הדוגמה, ההודעה מיוצגת כ-

**שכבה התעבורה (Transport Layer - TCP)** - בשלב זה מתווסף TCP Header, הכול בין היתר פורט מקור (Source Port), פורטיעד (Destination Port) ודגלים רלוונטיים. מידע זה מאפשר ניהול תקשורת אמינה בין הלקוח לשרת.

**שכבה הרשת (IP - Network Layer)** - בשלב זה מתווסף IP Header, המכיל את כתובות ה-IP של המקור והיעד. בפרויקט נעשה שימוש בכתובת Loopback (127.0.0.1), המדמה תקשורת מקומית בתוך אותו מחשב.

בסוף תהליך ה-Encapsulation ההודעה האריזה נשלחת כסימולציה תעבורת רשות אותה ניתן למכוד ולנתה באמצעות Wireshark.

### תהליך הילכידה ב-Wireshark:

כדי למכוד את התעבורה שנוצרה על ידי הסקייפ, הפעילנו את תוכנת Wireshark ונבחרנו ממושך Adapter for loopback traffic capture, כלומר תקשורת מקומית המבוצעת בתוך אותו מחשב.

לאחר בחירת הממשק הגדרנו מסנן תצוגה: `tcp.port == 12345`, מסנן זה שימש להציג התעבורה הרלוונטית לפרוייקט בלבד.

לאחר מכן הרכזנו את מחברת Jupyter, והתעבורה נלכדה בזמן אמיתי. בסיום ההרצה שמרנו את הילכידה בקובץ בפורמט `pcap`. לצורך ניתוח.

## ניתוח תעבורת Wireshark:

- התנועה שנדגמה ב-Wireshark משקפת תרחיש של "מבי סטום" בתקשורת TCP בין שני תהליכי מקומיים:

562 5.487409	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0
563 5.591926	127.0.0.1	127.0.0.1	TCP	59 [TCP Retransmission] 28402 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
564 5.592108	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0
565 5.699480	127.0.0.1	127.0.0.1	TCP	68 [TCP Retransmission] 28402 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=24
566 5.699633	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0
567 5.801465	127.0.0.1	127.0.0.1	TCP	55 [TCP Retransmission] 28402 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
568 5.801512	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0
569 5.904812	127.0.0.1	127.0.0.1	TCP	64 [TCP Retransmission] 28402 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=20
570 5.904913	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0
571 6.007550	127.0.0.1	127.0.0.1	TCP	59 [TCP Retransmission] 28402 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
572 6.007638	127.0.0.1	127.0.0.1	TCP	44 12345 → 28402 [RST] Seq=1 Win=0 Len=0

### מה רואים בתמונה?

- הניסיונן: כתובת המקור (כתובת 127.0.0.1) מנסה לשולח מידע מפורט 28402 לפורט 12345.
- התוקן: השולח מנסה להעביר פיסות מידע קטנות (בגודל 15, 20 או 24 בתים) ומשתמש בדגל **PSH** כדי לבקש מהצד השני לטפל במידע מייד.
- החסימה (RST): בכל פעם ששנשלח מידע, הצד מקבל (פורט 12345) עונה מיד עם הודעתת **RST**.
- המשמעות של RST היא שהחיבור סגור.
- שליחה מחדש (Retransmission): בגלל שהצד השולח לא מותר, הוא מנסה לשולח את אותו המידע שוב ושוב (מושפע-TCP Retransmission), ונדרשה שוב ושוב.

### למה זה קורה?

- אין לנו שרת ולכן אין תהליך שמאזין לפורט 12345, ולכן חוזרות הודעות RST.
- הלקוח לא מקבל סגירה מסודרת ולכן מנסה לבצע Retransmission, שליחת המידע שוב.

### הודעת אישור מהיעד אל הלוקוח על חיבור מוצלח והעברת הודעות תקינה:

0000	02 00 00 00 45 00 00 37 00 01 00 00 40 06 7c be	.....E...7 .....@.. ..
0010	7f 00 00 01 7f 00 00 01 6e f2 30 39 00 00 00 00	.....n..09
0020	00 00 00 00 50 18 20 00 3c d7 00 00 48 54 54 50	....P.. <...HTTP
0030	2f 31 2e 31 20 32 30 30 20 4f 4b	/1.1 200 OK

- ניתוח הנתונים של החבילה:

```
Protocol: TCP (6)
Header Checksum: 0xcbe [validation disabled]
[Header checksum status: Unverified]
Source Address: 127.0.0.1
Destination Address: 127.0.0.1
[Stream index: 1]
▼ Transmission Control Protocol, Src Port: 28402, Dst Port: 12345, Seq: 1, Ack: 1, Len: 15
  Source Port: 28402
  Destination Port: 12345
  [Stream index: 7]
  [Stream Packet Number: 11]
  > [Conversation completeness: Incomplete (40)]
  [TCP Segment Len: 15]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 0
  [Next Sequence Number: 16 (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: 0x3cd7 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
  Client continuous streams: 0
```

מה רואים בתמונה?

### 1. זהות השולח והمستقبل

- כתובות ה-IP: גם כתובת המקור וגם כתובת היעד הן 127.0.0.1. זה אומר שהתקשרות מתבצעת בתוך המחשב.
- פורט מקור (Source Port): 28402 זה הפורט שמתחל את ההתקשרות.
- פורט יעד (Destination Port): 12345 זה הפורט אליו נשלח המידע.

### 2. מצב התקשרות

- :Protocol TCP מציין את סוג הпрוטוקול בו ההודעות עוברות.
- :Flags מציין את קבלת המידע קודם.
- :ACK Push (PSH) מציין את קבלת המידע קודם.
- (Push): אומרים למערכת להעביר את המידע ישר לאפליקציה ולא לחייב שהבאפר תמלא, מאחר וזה צאט לנו רצימם שהעברת המידע תהיה מהירה ככל הניתן.

### 3. נתונים הרצף והמטרען

- :Acknowledgment Number הוא 1, מה שמעיד על כך שהצד השולח מחייב לקבל את הביטח הראשון מהצד השני.
- :TCP Payload גודל המידע שנשלח בחבילה זו הוא 15 בתים.

