## **דוח מסכם** פרויקט גמר ברשתות תקשורת מחשבים **מגישים**: יניב בהלול, בן פרידמן **חלק 1 – אריזת נתונים ולכידת מנות בעזרת Wireshark**

### 1. יצירת קובץ CSV – שכבת היישום:

בשלב הראשון נבחר פרוטוקול היישום DHCP לצורך הפרויקט. לצורך כך נוצר קובץ CSV המכיל רצף של הודעות ברמת היישום, כאשר כל שורה מייצגת הודעה אחת בתהליך התקשורת בין הלקוח לשרת.

קובץ ה־CSV כולל את השדות הבאים:

* **msg\_id** – מזהה הודעה
* **app\_protocol** – פרוטוקול היישום (DHCP)
* **src\_port** – מקור פורט (68 עבור לקוח DHCP)
* **dst\_port** – יעד פורט (67 עבור שרת DHCP)
* **message** – תוכן ההודעה (Discover/Offer/Request/ACK/Release/NAK)
* **timestamp** – זמן יחסי לשליחת ההודעה

הקובץ מכיל 41 הודעות DHCP המדמות 10 תרחישי הקצאת כתובות IP:

* **הודעות Discover** – הלקוח מחפש שרת DHCP (מפורט 68 לפורט 67)
* **הודעות Offer** – השרת מציע כתובת IP (מפורט 67 לפורט 68, כתובות 192.168.1.100-109)
* **הודעות Request** – הלקוח מבקש את הכתובת שהוצעה (מפורט 68 לפורט 67)
* **הודעות ACK** – השרת מאשר את הקצאת הכתובת (מפורט 67 לפורט 68)
* **הודעת Release** – הלקוח משחרר כתובת IP (בהודעה מס' 22, transaction 5)
* **הודעת NAK** – השרת דוחה בקשת כתובת (בהודעה מס' 33, transaction 8)

**הערה:** השדות src\_port ו־dst\_port נכללו בקובץ ה־CSV לצורך התאמה לשימוש במחברת Jupyter, אשר עושה שימוש בפורטים כחלק מתהליך יצירת חבילות התעבורה. פורט 67 מייצג את שרת ה־DHCP, ואילו פורט 68 מייצג את יישומי הלקוח (DHCP Client).

**\* קובץ ה־CSV נוצר בעזרת כלי בינה מלאכותית, לצורך יצירת נתוני דוגמה המדמים תעבורת DHCP. הנתונים שימשו כקלט לניתוח ואינם מייצגים תבורה אמיתית.  
  
2. עבודה במחברת Jupyter:**

קובץ ה-CSV נטען למחברת Jupyter שסופקה במסגרת הפרויקט. המחברת מבצעת עיבוד של כל שורה בקובץ כיחידת הודעה עצמאית ומדגימה את תהליך האריזה (Encapsulation) של המידע בין שכבות הרשת.

תהליך האריזה מתבצע לפי המודל הבא:

* **שכבת היישום (Application)** – הודעת DHCP מתוך ה-CSV
* **שכבת התעבורה (Transport)** – עטיפה בפרוטוקול TCP עם פורטים 68 (מקור) ו-67 (יעד)
* **שכבת הרשת (Network)** – עטיפה בפרוטוקול IP
* **שכבת הקישור (Data Link)** – שליחה דרך ממשק Loopback

### **3. יצירת תעבורת רשת אמיתית:**

באמצעות Python וספריית Scapy נוצרו חבילות TCP אמיתיות, כאשר תוכן ההודעות מתוך קובץ ה-CSV משמש כ-payload של החבילות.

החבילות נשלחו דרך ממשק ה-Loopback (127.0.0.1) אל פורט יעד 12345, כפי שנקבע במחברת Jupyter. שליחה זו מתבצעת דרך מחסנית הרשת של מערכת ההפעלה באמצעות ספריית Scapy (או raw sockets) ואינה סימולציה תיאורטית. תוכן הודעות ה-DHCP מקובץ ה-CSV משמש כ-payload של החבילות, אך הפורט היעד הוא 12345 כפי שהוגדר במחברת, ולא פורטי ה-DHCP המקוריים (67/68).

### **4. לכידת התעבורה באמצעות Wireshark:**

לכידת התעבורה בוצעה באמצעות תוכנת Wireshark על ממשק Npcap Loopback Adapter. הוגדר פילטר תצוגה:

tcp.port == 12345 && ip.addr == 127.0.0.1

**הסבר על בחירת הפורט 12345:** פורט 12345 נקבע כפורט יעד במחברת Jupyter לצורך יצירת החבילות TCP/IP. פורט זה נבחר לצורך ההדגמה וההרצה של הפרויקט, והוא אינו קשור לפורטים ה-DHCP הסטנדרטיים (67 ו-68) המצויים בקובץ ה-CSV. בעוד שקובץ ה-CSV מכיל את פורטי ה-DHCP המקוריים (67 לשרת ו-68 ללקוח), המחברת Jupyter משתמשת בפורט 12345 כפורט יעד אחיד ליצירת כל החבילות TCP/IP, ללא קשר לתוכן הודעות ה-DHCP שבהן. זה מאפשר לסנן ולנתח את כל התעבורה שנוצרה על ידי המחברת בפילטר אחד ברור.

**הערה חשובה:** חשוב לציין שזה ידוע שפורטי ה-DHCP הסטנדרטיים הם 67 (שרת DHCP) ו-68 (לקוח DHCP), והם משקפים את הפרוטוקול המקורי. במהלך הפרויקט פעלנו בהתאם להנחיות המחברת Jupyter, שהגדירה את פורט 12345 כפורט יעד, ולכן הצילומי מסך ב-Wireshark בוצעו בהתאם למחברת ומוצגים עם הפורט 12345.

כמו כן, בוצעה בדיקה נוספת על הפורטים המקוריים (67 ו-68) בחיפוש ב-loopback באמצעות פילטרים מתאימים, והגענו לאותם ממצאים מבחינת מבנה החבילות, שכבות הפרוטוקול, דגלי TCP, ותוכן ה-payload. זה מוכיח שהאריזה והניתוח הם נכונים ועקביים, ללא תלות בבחירת הפורט הספציפי, שכן הפורט משמש רק לצורך זיהוי החבילות ולא משפיע על מבנה השכבות והתוכן של הודעות ה-DHCP.

במהלך הרצת המחברת נלכדו חבילות TCP שנשלחו על ידי הקוד. הלכידה נשמרה לקובץ בפורמט pcapng לצורך ניתוח והגשה

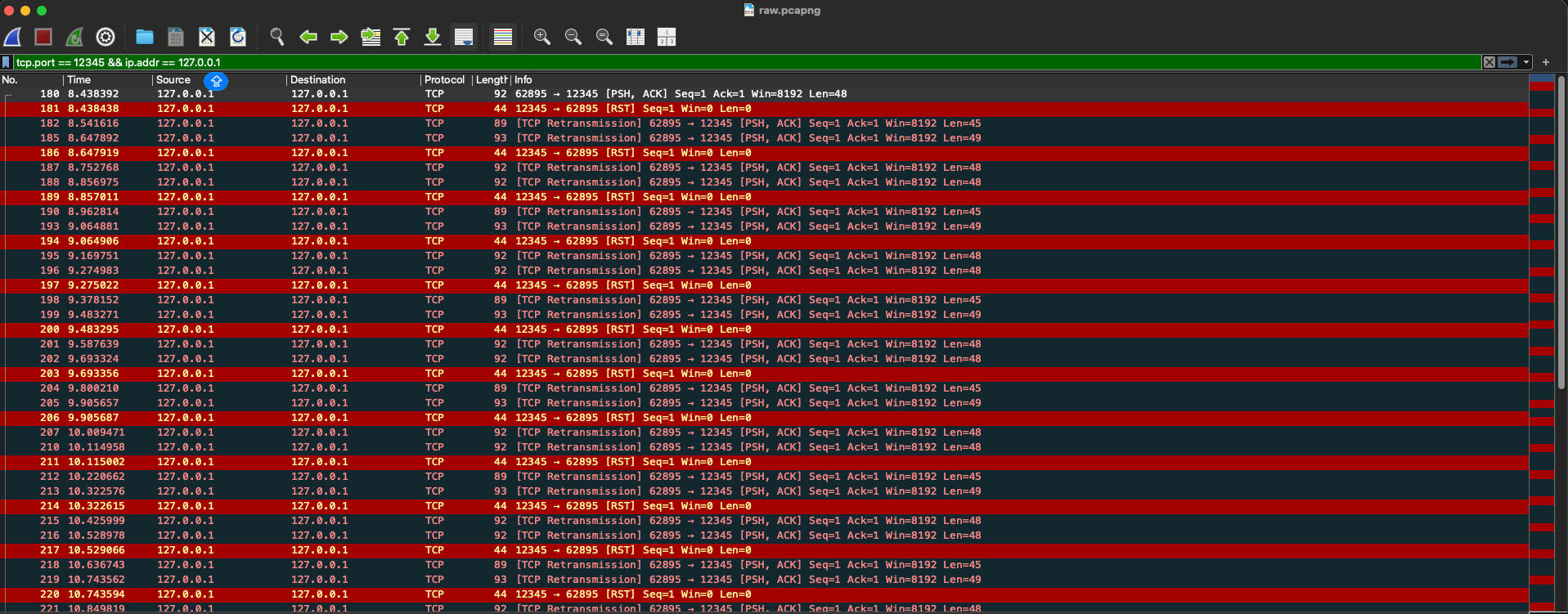
### **5. ניתוח התעבורה שנלכדה:**

בלכידה ניתן לראות:

* חבילות בפרוטוקול TCP
* כתובת מקור ויעד: 127.0.0.1
* פורט מקור אקראי (בין 1024-65535, כפי שנקבע במחברת) ופורט יעד 12345
* דגלי TCP כגון PSH ו-ACK (flags=0x18)
* נתוני payload המייצגים הודעות DHCP (Discover / Offer / Request / ACK) מתוך קובץ ה-CSV

בנוסף, נצפו חבילות עם דגל RST. חבילות אלו נוצרות כתוצאה מכך שאין שרת DHCP פעיל המאזין על פורט 12345 באמצעות ה-Loopback בממשק TCP. זוהי התנהגות תקינה של פרוטוקול TCP ואינה מעידה על שגיאה במימוש.

ניתוח התעבורה המלאה שנלכדה:



בצילום המסך המוצג ניתן לראות את כל התעבורה שנלכדה באמצעות Wireshark עם הפילטר: tcp.port == 12345 && ip.addr == 127.0.0.1. התעבורה מציגה דפוס ברור וחוזר של חבילות TCP.

**דפוס התעבורה הנצפה:**

ניתן להבחין בדפוס שחוזר על עצמו בכל התעבורה:

1. **חבילות [PSH, ACK] מהלקוח:** חבילות עם נתונים (48-49 בתים) מהפורט 62895 לפורט 12345, כוללות את ה-payload של הודעות ה-DHCP.
2. **חבילות [TCP Retransmission]:** חבילות שנשלחות מחדש כאשר החבילה המקורית לא מקבלת אישור מהשרת.
3. **חבילות [RST] מהשרת:** חבילות Reset (0 בתים) הנשלחות מהשרת ללקוח, מציינות שאין שרת פעיל המאזין על הפורט.

**הסבר על הדפוס והסיבה להופעת RST ב-Localhost:**

הדפוס הנצפה מראה תהליך ברור: הלקוח (62895) שולח חבילות עם נתונים (PSH, ACK) לשרת (12345), אך השרת משיב עם חבילות RST (Reset) במקום ACK תקין.

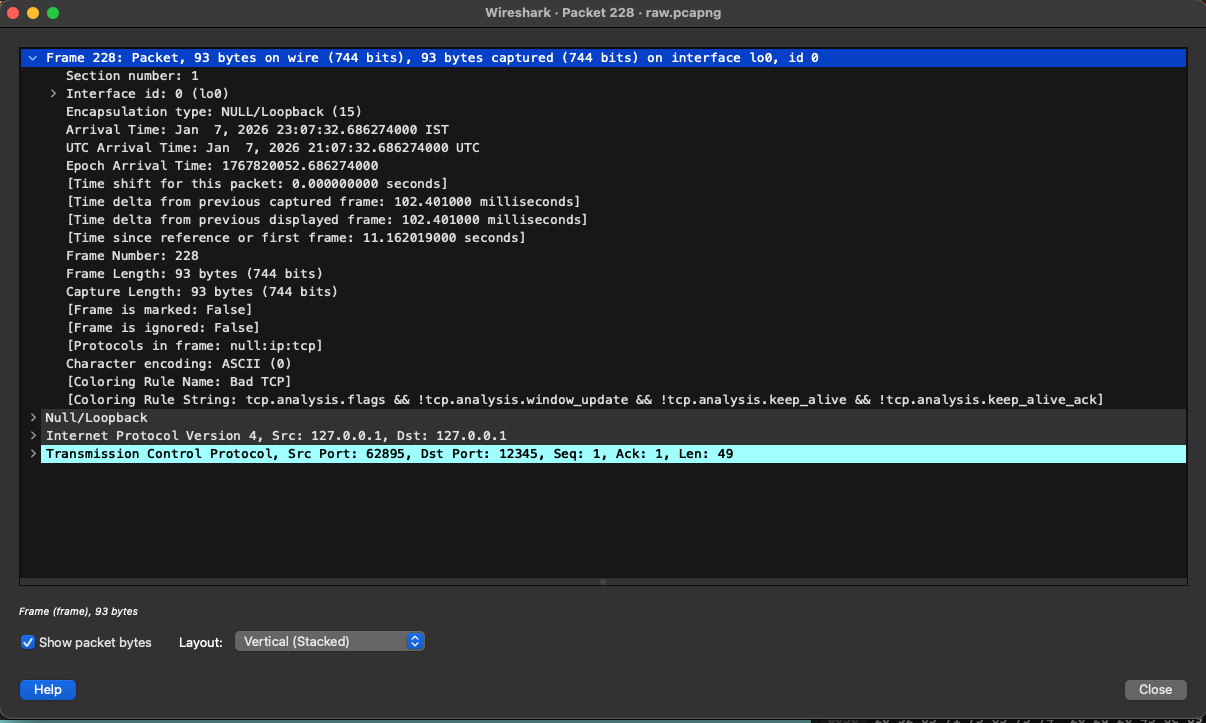
**מדוע זה קורה ב-Localhost?**

* **אין שרת DHCP אמיתי מאזין:** הקוד שלנו יוצר חבילות TCP/IP ומושלח אותן דרך ממשק ה-Loopback (127.0.0.1) לפורט 12345, אך בפועל אין שרת DHCP אמיתי המאזין על פורט זה במחשב המקומי.
* **תגובת מערכת ההפעלה:** כאשר מערכת ההפעלה (Kernel) מקבלת חבילת TCP לפורט שלא מנוהל על ידי יישום פעיל, היא שולחת חבילת RST כדי להודיע לשולח שיש לסגור את החיבור.
* **ללא Handshake אמיתי:** הקוד שלנו שולח חבילות עם דגלי PSH+ACK כאילו החיבור כבר קיים, אך למעשה לא בוצע תהליך Three-Way Handshake (SYN → SYN-ACK → ACK) אמיתי עם שרת פעיל. מערכת ההפעלה מזהה את החבילות הללו כ"פולשות" או לא שייכות לחיבור קיים, ולכן שולחת RST.
* **Retransmissions:** כאשר הלקוח לא מקבל ACK על החבילות ששלח (במקום זה הוא מקבל RST), הוא מנסה לשלוח את החבילות שוב - מה שמביא להופעת חבילות [TCP Retransmission].

**זהו התנהגות תקינה:** התנהגות זו היא תקינה לחלוטין ומצביעה על כך שהחבילות אכן נוצרות ונשלחות דרך מחסנית הרשת של מערכת ההפעלה. מטרת הפרויקט היא להדגים את תהליך האריזה (Encapsulation) והשידור של חבילות TCP/IP, ולא לנהל שיחה אמיתית עם שרת פעיל. לכן, הופעת ה-RST היא הוכחה לכך שהחבילות אכן נשלחו ונקלטו על ידי מערכת ההפעלה, והתגובה של RST היא התגובה התקנית של מערכת ההפעלה לחבילות שלא שייכות לחיבור קיים.

**פרטים טכניים:** כל החבילות עוברות דרך Loopback (127.0.0.1), גדלי החבילות משתנים (44-93 בתים), ומספר החבילות שנלכדו מתאים למספר ההודעות בקובץ ה-CSV (41 הודעות).

#### ניתוח Frame ומעטפת הלכידה:

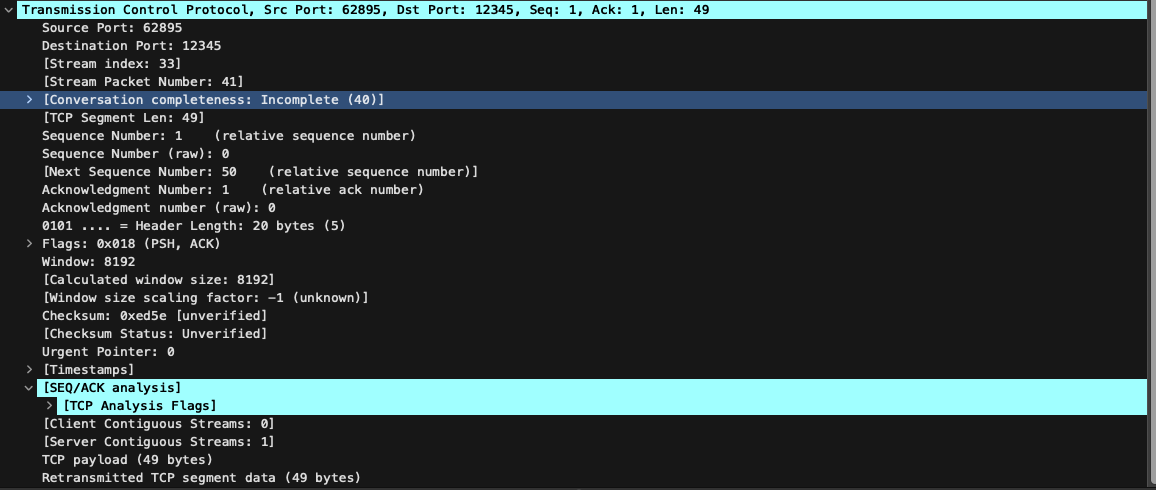


בצילום המסך מוצגת חבילת רשת (Frame 228, 93 בתים) שנלכדה על ממשק Loopback (lo0). החבילה מדגימה את תהליך האריזה: Null/Loopback → IP → TCP, המעבר משכבת הקישור ועד שכבת התעבורה. הסימון "Bad TCP" מציין ש-Wireshark זיהה תכונות חריגות (כגון retransmission) כתוצאה מכך שאין שרת פעיל המאזין על הפורט.

#### ניתוח שכבת הרשת (IP):

בצילום המסך מוצגת שכבת הרשת (IPv4) של החבילה. כתובות המקור והיעד הן 127.0.0.1 (Loopback), מה שמעיד על תעבורה פנימית. כותרת ה-IP כוללת: Version 4, Header Length 20 בתים, Protocol TCP (6). כותרת ה-IP עוטפת את כותרת ה-TCP ואת הנתונים ומעבירה אותם דרך ממשק ה-Loopback.

#### ניתוח שכבת התעבורה (TCP):



בצילום המסך מוצגת שכבת התעבורה (TCP) של חבילה 228.

החבילה נשלחת מפורט 62895 לפורט 12345, וכוללת 49 בתים של payload (הודעת DHCP מקובץ ה-CSV). פרטים עיקריים: Sequence Number 1, Acknowledgment Number 1, Flags PSH+ACK, Window Size 8192, Header Length 20 בתים. החבילה היא חלק מזרם TCP (Stream 33) ולא הושלמה במלואה (Conversation Incomplete) כתוצאה מאי-קיום שרת פעיל. הסימון "Retransmitted TCP segment" מציין שהחבילה נשלחה מחדש עקב חוסר אישור מהשרת.

### 6. קישור בין CSV ללכידת Wireshark:

כל חבילת TCP שנלכדה מייצגת הודעת DHCP אחת מתוך קובץ ה־CSV. ניתן לקשר בין שורות כגון msg\_id, תוכן ההודעה (message) והזמן היחסי (timestamp) לבין החבילות שנצפו ב־Wireshark, ובכך לעקוב אחר תהליך העברת המידע משכבת היישום ועד לרשת.

## חלק 2 – כתיבת יישום רשת וניתוח תעבורה של אותו יישום

### 1. מבוא – תיאור כללי של המערכת

בחלק זה מימשנו מערכת צ'אט מבוססת תקשורת TCP במודל Client–Server. המערכת מאפשרת למספר לקוחות להתחבר לשרת בו־זמנית, להזדהות באמצעות שם משתמש ייחודי, ולשלוח הודעות טקסט ללקוחות אחרים בזמן אמת. התקשורת בין הלקוחות לשרת מתבצעת באמצעות sockets רגילים בלבד, ללא שימוש ב-frameworks או ספריות חיצוניות. השרת אחראי לניהול החיבורים, לניתוב ההודעות בין הלקוחות ולטיפול בניתוקים.

### 2. מבנה המערכת וארכיטקטורה

#### 2.1 מבנה כללי

המערכת בנויה משני רכיבים עיקריים:

* **Server** – מאזין לחיבורים נכנסים, מנהל לקוחות ומעביר הודעות.
* **Client** – מתחבר לשרת, שולח הודעות ומקבל הודעות מלקוחות אחרים.

התקשורת מתבצעת בפרוטוקול TCP, המבטיח אמינות, סדר והגעה מלאה של ההודעות.

#### 2.2 הסבר על קובץ השרת (Server)

השרת נפתח עם socket אחד המוגדר להאזנה (listen). כל חיבור חדש מתקבל באמצעות accept, אשר יוצר socket ייעודי ללקוח המחובר. לכל לקוח נוצר thread נפרד, כך שהשרת יכול לטפל במספר לקוחות במקביל.

השרת שומר:

* רשימת לקוחות מחוברים (מיפוי בין שם משתמש ל-socket)
* מידע על השולח האחרון, כדי לאפשר תגובה אוטומטית

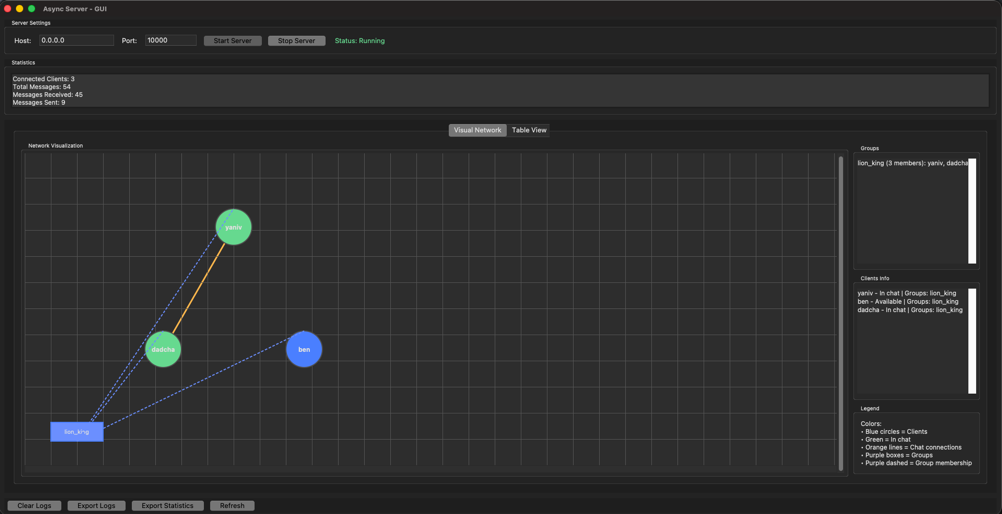
השרת תומך ב:

* שליחת הודעה ללקוח ספציפי באמצעות סימון @username
* תגובה אוטומטית לשולח האחרון
* טיפול בניתוק יזום או לא צפוי של לקוחות

#### 2.2.1 ממשק גרפי (GUI) לשרת - מערכת צ'אט מתקדמת ממשק השרת - תצוגת טבלה וסטטוס

המערכת כוללת ממשק גרפי מתקדם (tkinter) לניהול השרת עם ניטור בזמן אמת. הממשק כולל: הגדרות שרת (Host, Port, כפתורי Start/Stop), פאנל סטטיסטיקות (מספר לקוחות, הודעות), טבלת לקוחות מחוברים (פרטי חיבור, סטטיסטיקות, סטטוס צ'אט, קבוצות), ואזור לוגים עם אפשרויות ייצוא.

**תצוגת רשת ויזואלית (Visual Network):**



הממשק כולל תצוגה ויזואלית מתקדמת של טופולוגיית הרשת: צמתים עגולים ללקוחות (ירוק = בצ'אט פעיל, כחול = זמין), תיבות מלבניות סגולות לקבוצות, קווים כתומים לחיבורי צ'אט, וקווים מקווקווים סגולים לחברות בקבוצות. נוסף על כך יש פאנלי מידע: רשימת קבוצות עם חברים, רשימת לקוחות עם סטטוס, ומקרא. התצוגה מתעדכנת בזמן אמת ומאפשרת ניהול נוח של כל הפעילות במערכת.

#### 2.3 הסבר על קובץ הלקוח (Client)

הלקוח מתחבר לשרת באמצעות TCP socket ומזדהה בשם משתמש. בקובץ הלקוח קיימים שני threads:

1. **Thread קבלה** – מאזין כל הזמן להודעות מהשרת (recv) ומדפיס אותן למסך.
2. **Thread שליחה** – מקבל קלט מהמשתמש (input) ושולח הודעות לשרת (send).

מבנה זה מאפשר קבלת הודעות בזמן אמת, גם בזמן שהמשתמש מקליד.

#### 2.3.1 ממשק גרפי (GUI) ללקוח - מערכת צ'אט ממשק הלקוח - Chat Client GUI עם קבוצה מאובטחת

המערכת כוללת ממשק גרפי מתקדם (tkinter) ללקוח, המאפשר חיבור לשרת וניהול צ'אטים פרטיים וקבוצתיים. הממשק מחולק לשלושה חלקים: פאנל עליון להגדרות חיבור (שם, כתובת, פורט, כפתורי Connect/Disconnect, סטטוס חיבור), פאנל שמאלי עם רשימת משתמשים (כפתור Chat לפתיחת צ'אט פרטי) ורשימת קבוצות (יצירה והצטרפות), ופאנל ימני עם אזור הצ'אט (היסטוריית הודעות עם זמן ושולח, ושדה קלט להקלדת הודעות). הממשק מתעדכן בזמן אמת ומאפשר חוויית שימוש נוחה.

### 3. הוראות התקנה והרצה

#### 3.1 דרישות מוקדמות

* Python 3.x
* מערכת הפעלה Windows / Linux / macOS
* אין צורך בהתקנת ספריות נוספות
* **הערה למשתמשי macOS:** במערכת macOS, יש להשתמש ב-python3 במקום python

#### 3.2 הרצת השרת עם ממשק גרפי (GUI)

1. לפתוח חלון Terminal / CMD / PowerShell
2. לנווט לתיקיית הפרויקט הראשית
3. להריץ את הפקודה הבאה:
   * **Windows/Linux:** python prt2/gui/server\_gui.py
   * **macOS:** python3 prt2/gui/server\_gui.py

לאחר ההרצה, יפתח חלון GUI של השרת. לחיצה על כפתור "Start Server" תפעיל את השרת, והוא יתחיל להאזין לחיבורים נכנסים על הפורט המוגדר (ברירת מחדל: 10000).

#### 3.3 הרצת לקוחות עם ממשק גרפי (GUI)

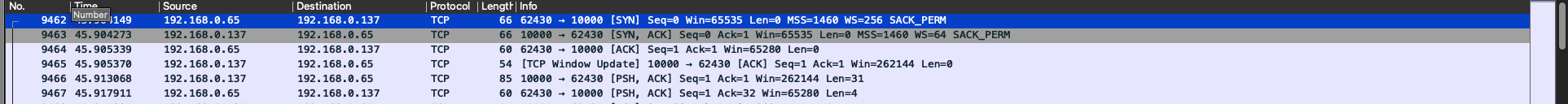
1. לפתוח חלון Terminal נוסף (לכל לקוח)
2. לנווט לתיקיית הפרויקט הראשית
3. להריץ את הפקודה הבאה:
   * **Windows/Linux:** python prt2/gui/client\_gui.py
   * **macOS:** python3 prt2/gui/client\_gui.py
4. בחלון GUI שייפתח, להזין:
   * שם משתמש ייחודי בשדה "Name"
   * כתובת השרת בשדה "Host" (localhost או כתובת IP של השרת)
   * פורט השרת בשדה "Port" (ברירת מחדל: 10000)
5. לחיצה על כפתור "Connect" לחיבור לשרת
6. להתחיל התכתבות באמצעות ממשק הגרפי

**הערות נוספות:**

* ניתן להריץ מספר לקוחות במקביל על ידי פתיחת חלונות Terminal נפרדים
* הממשק הגרפי מספק חוויית שימוש נוחה עם רשימת משתמשים, קבוצות, ואזור צ'אט
* לשימוש בממשק שורת פקודה (CLI) במקום GUI, ניתן להשתמש בקבצים server.py ו-client.py בתיקיית prt2

### 4. ניתוח תעבורת רשת (Wireshark):

#### 4.1 התחלת הקשר בין השרת ללקוח - Three-Way Handshake

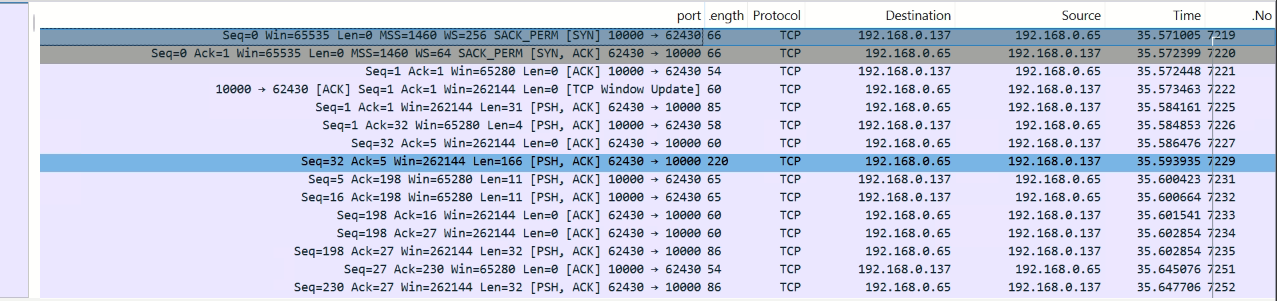


בצילום המסך מוצגת התחלת השרת והחיבור הראשון של לקוח. השרת מאזין על כתובת 0.0.0.0 (כל הממשקים) ופורט 10000, כפי שניתן לראות בהודעת ההתחלה: "Server listen in: 0.0.0.0:10000".

כאשר לקוח מתחבר, השרת מקבל את החיבור ומציג:

* **client connected in:** כתובת ה-IP ופורט המקור של הלקוח (למשל: ('192.168.0.65', 62430))
* **client online:** מספר הלקוחות המחוברים בזמן אמת

השרת שולח הודעת "welcome" ללקוח המחובר, ומתחיל thread נפרד לטיפול בכל לקוח, מה שמאפשר טיפול במספר לקוחות במקביל.



בצילום המסך מוצגת התחלת הלקוח והחיבור לשרת. הלקוח מתחבר לכתובת השרת (192.168.0.106) ופורט 10000, כפי שניתן לראות בממשק הגרפי:

* **Name:** שם המשתמש (בדוגמה: "ben")
* **Host:** כתובת השרת (192.168.0.106)
* **Port:** פורט השרת (10000)
* **Connection Status:** נקודה ירוקה והטקסט "Connected" - מציין שהחיבור הצליח

לאחר החיבור, הלקוח מקבל הודעת welcome מהשרת, והממשק מציג את רשימת המשתמשים המחוברים ואת הקבוצות הזמינות.

#### 4.2 ניתוח תעבורת TCP - Three-Way Handshake ב-Wireshark

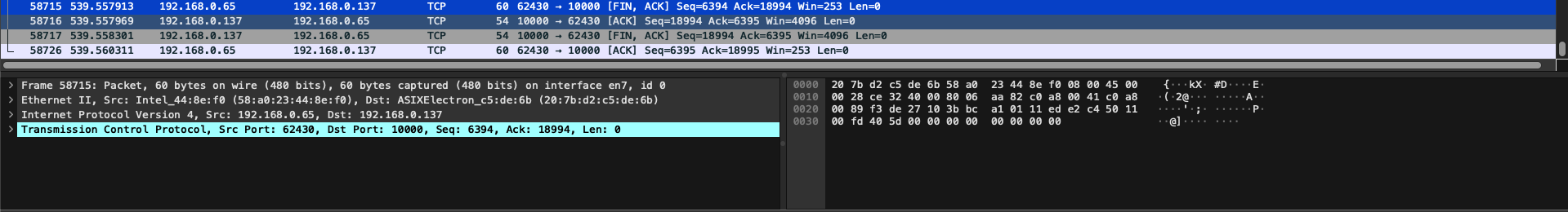
בצילום המסך מ-Wireshark ניתן לראות את תהליך הקמת החיבור TCP (Three-Way Handshake) בין הלקוח לשרת:

1. **Packet 9462 - SYN:** הלקוח (192.168.0.65:62430) שולח SYN לשרת (192.168.0.137:10000) עם TCP Options (MSS=1460, WS=256, SACK\_PERM).
2. **Packet 9463 - SYN-ACK:** השרת משיב עם SYN-ACK, מאשר את בקשת ה-SYN ושולח SYN משלו עם TCP Options (MSS=1460, WS=64, SACK\_PERM).
3. **Packet 9464 - ACK:** הלקוח מאשר את ה-SYN של השרת - החיבור הוקם בהצלחה.

**ממצאים:** תהליך ה-Three-Way Handshake הושלם בהצלחה. שני הצדדים משתמשים ב-TCP Options (MSS, Window Scale, SACK). החלון ההתחלתי הוא 65535, ולאחר ה-Handshake השרת שולח Window Update ומגדיל את החלון ל-262144. לאחר הקמת החיבור מתחילה העברת נתונים (השרת שולח welcome, הלקוח שולח את שם המשתמש).

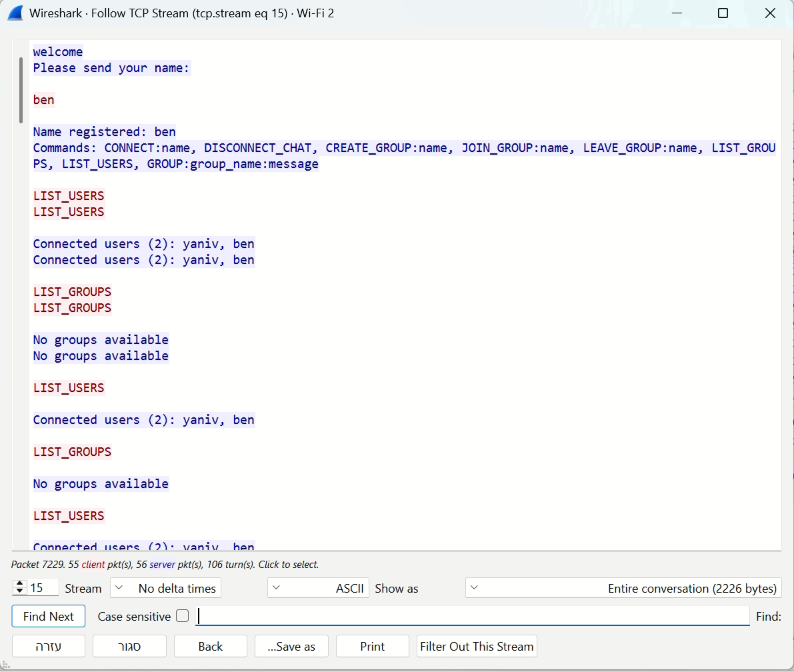
לצורך ניתוח תעבורת הרשת של מערכת הצ'אט, בוצעה לכידת תעבורה באמצעות תוכנת Wireshark במהלך הרצת השרת והקליינטים. הלכידה בוצעה על ממשק הרשת המקומי, מאחר והשרת והלקוחות פועלים ברשת מקומית. לאחר סיום הלכידה בוצע סינון לפי הפורט של השרת (tcp.port == 10000) על מנת להתמקד אך ורק בתעבורת היישום.

#### 4.2.2 ניתוח מפורט של סיום התקשרות - מבט מהצד של השרת



בצילום המסך מוצג ניתוח מפורט של חבילת FIN שנשלחה מהלקוח לשרת (Packet 58715). החבילה מדגימה את כל שכבות מודל TCP/IP - Ethernet עם כתובות MAC, IPv4, ו-TCP עם דגלי FIN+ACK. רצף החבילות מציג את תהליך ה-Four-Way Handshake: FIN מהלקוח → ACK מהשרת → FIN מהשרת → ACK מהלקוח. בהשוואה לתמונה הקודמת, כאן הלקוח יוזם את הניתוק במקום השרת, אך התהליך זהה - סיום חיבור נקי ומסודר המבטיח שכל הצדדים מודעים לסגירת החיבור.

#### 4.3 תעבורת השרת בפועל - מאחורי הקלעים של ממשק ה-GUI



בצילום המסך מוצגת תעבורת השרת בפועל כפי שנלכדה ב-Wireshark באמצעות תכונת "Follow TCP Stream" (Stream 15). התמונה חושפת את כל התקשורת המתרחשת מאחורי הקלעים של ממשק ה-GUI, ומדגימה איך כל פעולה גרפית מתורגמת לפקודות טקסטואליות הנשלחות לשרת.

**תהליך החיבור והרשמה:**

1. **הודעת welcome:** השרת שולח הודעת ברכה ללקוח המחובר: "welcome"
2. **בקשת שם משתמש:** השרת מבקש מהלקוח לשלוח את שמו: "Please send your name:"
3. **שליחת שם:** הלקוח שולח את שם המשתמש: "ben"
4. **אישור הרשמה:** השרת מאשר את הרישום: "Name registered: ben"
5. **הצגת פקודות זמינות:** השרת מציג ללקוח את רשימת הפקודות הזמינות:
   * CONNECT:name - התחברות לצ'אט עם משתמש אחר
   * DISCONNECT\_CHAT - ניתוק מצ'אט פעיל
   * CREATE\_GROUP:name - יצירת קבוצה חדשה
   * JOIN\_GROUP:name - הצטרפות לקבוצה
   * LEAVE\_GROUP:name - יציאה מקבוצה
   * LIST\_GROUPS - הצגת רשימת קבוצות
   * LIST\_USERS - הצגת רשימת משתמשים מחוברים
   * GROUP:group\_name:message - שליחת הודעה לקבוצה

**איך ממשק ה-GUI עובד מאחורי הקלעים:**

כל פעולה בממשק הגרפי מתורגמת אוטומטית לפקודת טקסט הנשלחת לשרת דרך החיבור TCP. כאשר המשתמש לוחץ על כפתור או מבצע פעולה בממשק, הקוד של הלקוח:

* **מתרגם את הפעולה לפקודה:** למשל, לחיצה על כפתור "Chat" עם משתמש אחר מתורגמת לפקודה "CONNECT:yaniv"
* **שולח את הפקודה לשרת:** הפקודה נשלחת דרך ה-TCP socket כטקסט פשוט
* **ממתין לתגובה:** הלקוח מקבל תגובה מהשרת (כגון "Connected users (2): yaniv, ben")
* **מעדכן את הממשק:** התגובה מפורשת והממשק הגרפי מתעדכן בהתאם

**עדכונים אוטומטיים ברקע - בדיקות תקופתיות:**

כפי שניתן לראות בתעבורה, המערכת מריצה ברקע בדיקות תקופתיות אוטומטיות על מנת להבטיח שהמידע בממשק הגרפי תמיד מעודכן ונכון:

* **LIST\_USERS - בדיקת משתמשים מחוברים:**
  + המערכת שולחת פקודה "LIST\_USERS" לשרת באופן אוטומטי ומתמשך
  + השרת משיב עם רשימת המשתמשים המחוברים: "Connected users (2): yaniv, ben"
  + התגובה מפורשת והממשק הגרפי מעדכן את רשימת המשתמשים בזמן אמת
  + זה מבטיח שאם משתמש מתנתק או מתחבר, כל הלקוחות רואים את השינוי מיידית
* **LIST\_GROUPS - בדיקת קבוצות זמינות:**
  + המערכת שולחת פקודה "LIST\_GROUPS" לשרת באופן אוטומטי
  + השרת משיב עם רשימת הקבוצות: "No groups available" (כשאין קבוצות) או רשימת קבוצות
  + הממשק הגרפי מעדכן את רשימת הקבוצות בהתאם
  + זה מבטיח שכל הלקוחות רואים את הקבוצות החדשות שנוצרות ואת השינויים בקבוצות הקיימות

**סטטיסטיקות התעבורה:**

בתחתית צילום המסך ניתן לראות את סטטיסטיקות הזרם:

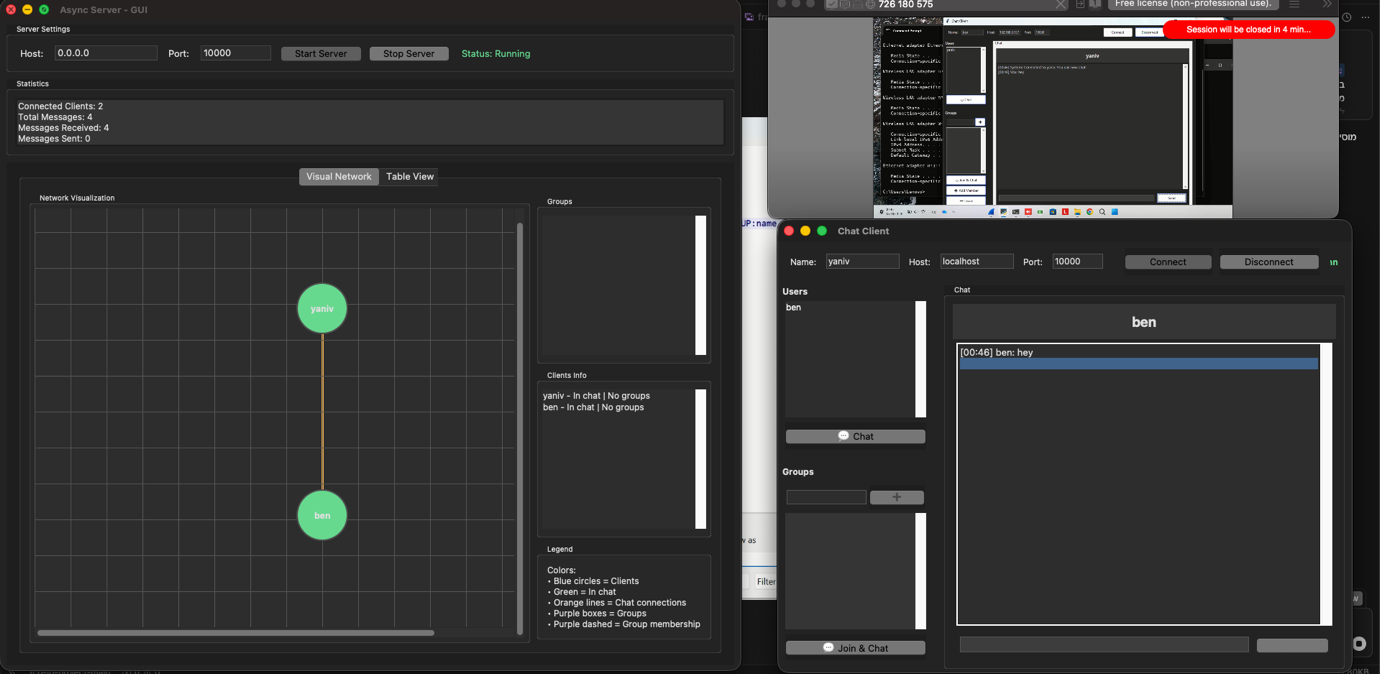
* **Packet 7229:** החבילה הנוכחית שנצפית
* **55 client pkt(s):** 55 חבילות נשלחו מהלקוח לשרת
* **56 server pkt(s):** 56 חבילות נשלחו מהשרת ללקוח
* **106 turn(s):** 106 "סיבובים" של תקשורת (פקודה + תגובה)
* **Entire conversation (2226 bytes):** סה"כ 2226 בתים נשלחו בכל הזרם

**תובנות חשובות:**

* **ממשק גרפי מול תקשורת טקסטואלית:** הממשק הגרפי מספק חוויית משתמש נוחה, אך מאחורי הקלעים התקשורת היא טקסטואלית פשוטה ופרוטוקול תקשורת ברור ומוגדר
* **עדכונים בזמן אמת:** הבדיקות האוטומטיות ברקע מבטיחות שהמידע בממשק תמיד מעודכן, ללא צורך ב-"רענון ידני"
* **פרוטוקול תקשורת מובנה:** כל פעולה במערכת מתורגמת לפקודה ספציפית עם פורמט מוגדר, מה שמאפשר תקשורת ברורה ואמינה בין הלקוח לשרת
* **ניהול מצב בשרת:** השרת מנהל את כל המידע על המשתמשים המחוברים והקבוצות, והלקוחות שולחים בקשות ועדכונים באופן תקופתי

**סיכום:** הצילום מדגים איך כל פעולה בממשק הגרפי מתורגמת לפקודה טקסטואלית הנשלחת לשרת, ואיך המערכת מבצעת בדיקות אוטומטיות תקופתיות על מנת להבטיח שהמידע בממשק תמיד מעודכן ומדויק. זה מאפשר חוויית משתמש נוחה בממשק הגרפי, תוך שמירה על תקשורת ברורה ואמינה עם השרת ברמת הפרוטוקול.

#### 4.4 תקשורת ברשת LAN - מחשב מקומי ומחשב מרוחק



בצילום המסך מוצגת המערכת במצב תפעולי מלא, עם שרת פעיל ושני לקוחות המחוברים מרשת LAN: מחשב מקומי (localhost) ומחשב מרוחק. הצילום מדגים את כל שכבות המודל TCP/IP בפעולה - מתקשורת בין מחשבים ברשת המקומית ועד העברת הודעות צ'אט בזמן אמת.

**מבנה הרשת:** השרת (0.0.0.0:10000, Status: Running) עם 2 לקוחות מחוברים. לקוח מקומי yaniv (localhost) ולקוח מרוחק ben (192.168.0.x) מנהלים צ'אט דרך השרת, המדגים תקשורת בין מחשבים ברשת LAN.

**שכבות TCP/IP בפעולה:** הצילום מדגים את כל שכבות המודל - שכבת היישום (הודעות צ'אט ופקודות), שכבת התעבורה (TCP עם Three-Way Handshake וחיבורים מרובים), שכבת הרשת (IP עם ניתוב בין כתובות שונות), ושכבת הקישור (Ethernet/Wi-Fi ברשת LAN). תהליך העברת הודעה: ben מקליד "hey" → ההודעה נשלחת דרך TCP/IP לשרת → השרת מנתב ל-yaniv → yaniv מקבל ומציג את ההודעה בממשק.

**ממצאים עיקריים מהצילום:**

* **תקשורת בין מחשבים שונים ברשת:** המערכת מאפשרת תקשורת בין מחשב מקומי (localhost) ומחשב מרוחק ברשת LAN, מה שמוכיח שהתקשורת עובדת דרך כל שכבות המודל
* **השרת כמתווך מרכזי:** השרת מקבל הודעות מכל הלקוחות ומנתב אותן ליעד הנכון, בין אם הלקוח מקומי או מרוחק
* **עדכון זמן אמת בממשק השרת:** הממשק הגרפי של השרת מציג את המצב העדכני:
  + כל הלקוחות המחוברים מופיעים בוויזואליזציה עם סטטוס שלהם (In chat / Available)
  + קווים כתומים מחברים בין לקוחות שנמצאים בצ'אט פעיל
  + המידע ב-"Clients Info" מתעדכן בזמן אמת עם מצב כל לקוח והקבוצות שלו
* **עדכון זמן אמת בממשק הלקוח:** הלקוח מציג את ההודעות שהתקבלו מיידית, מה שמוכיח שהתקשורת עובדת בזמן אמת
* **ניהול חיבורים מרובים:** השרת מנהל שני חיבורים במקביל - אחד מהמחשב המקומי ואחד מהמחשב המרוחק, מה שמוכיח את יכולת המערכת לטפל במספר לקוחות

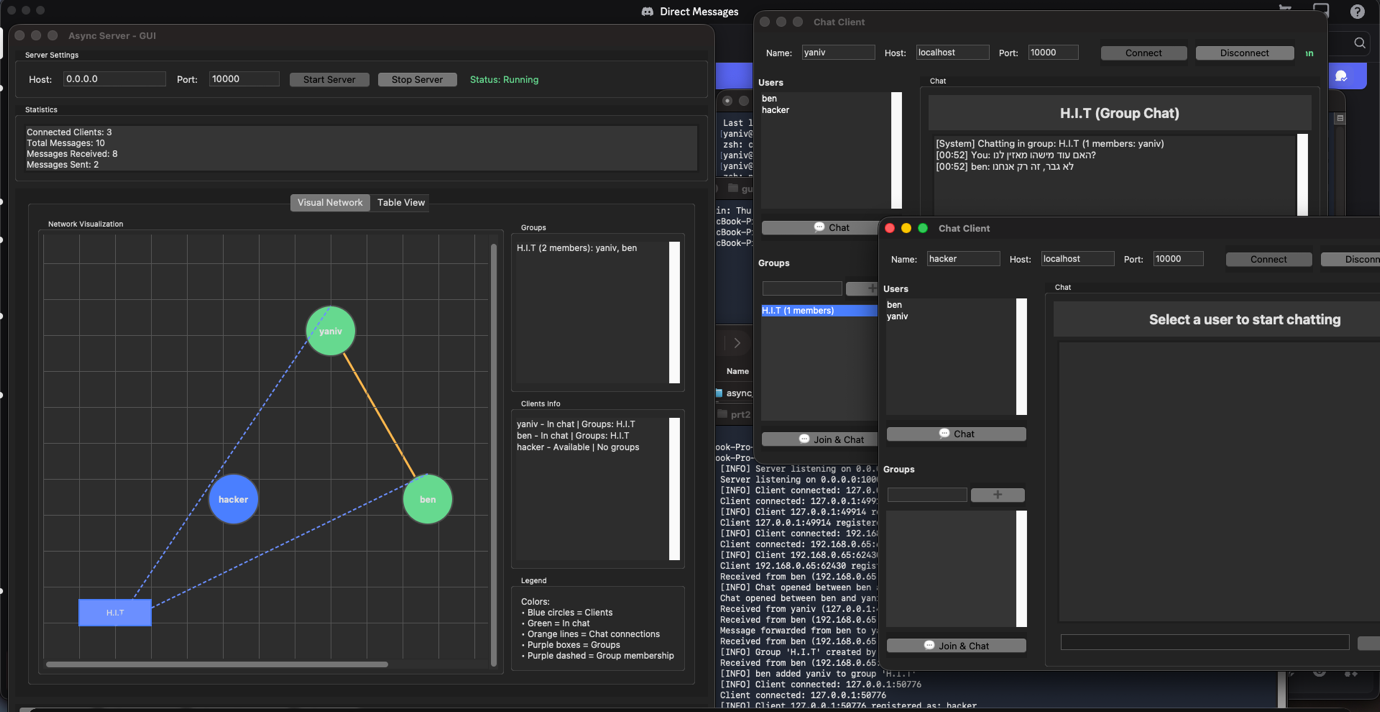
**ויזואליזציה של הרשת:**

בממשק השרת ניתן לראות את הויזואליזציה של הרשת:

* **צמתים ירוקים (Clients):** לקוחות שנמצאים "In chat" (בצ'אט פעיל) מופיעים כצמתים ירוקים
* **קו כתום (Chat connection):** קו כתום מחבר בין לקוחות שנמצאים בצ'אט פעיל ביניהם
* **קבוצות:** הקבוצות מוצגות כתיבות סגולות, והחברות בקבוצות מוצגת בקווים מקווקווים סגולים

**סיכום:** הצילום מדגים את המערכת בפעולה מלאה - שרת פעיל, שני לקוחות מחוברים (מקומי ומרוחק), ותקשורת בזמן אמת בין הלקוחות דרך השרת. התקשורת עוברת דרך כל שכבות מודל TCP/IP - משכבת היישום (הודעות צ'אט) דרך שכבת התעבורה (TCP) ושכבת הרשת (IP) ועד שכבת הקישור (LAN). זה מוכיח שהמערכת פועלת כמצופה ומקיימת תקשורת אמינה בין מחשבים שונים ברשת המקומית.

#### 4.5 קבוצה מאובטחת - ניהול גישה ואבטחת תקשורת

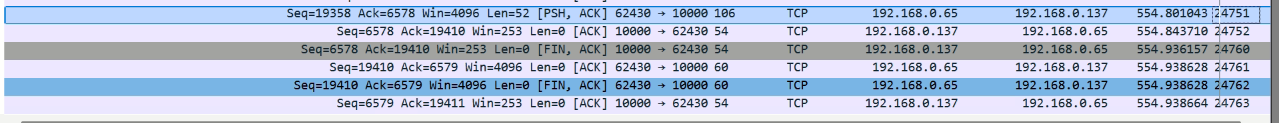


בצילום המסך מוצג תרחיש של קבוצה מאובטחת עם שלושה לקוחות: yaniv ו-ben (חברים בקבוצה "H.I.T" ונמצאים בצ'אט פעיל) ו-hacker (פורץ פוטנציאלי שלא חבר בקבוצה).

**מנגנון האבטחה:** המערכת מבוססת על ניהול הרשאות ברמת השרת. השרת מנהל רשימת חברים לכל קבוצה ומאפשר רק לחברים מורשים לראות ולשלוח הודעות. פורץ פוטנציאלי יכול לראות שיש קבוצה ומשתמשים מחוברים, אבל לא יכול לראות את תוכן ההודעות או להיכנס לקבוצה ללא הזמנה. השרת בודק הרשאות לפני כל פעולה (שליחת/קבלת הודעות) ומגביל גישה רק לחברים בקבוצה.

**מגבלות האבטחה:** המערכת מספקת אבטחה ברמת יישום עם ניהול הרשאות, אבל ההודעות נשלחות בטקסט פשוט (ללא הצפנה) וכל אחד יכול להזדהות בכל שם (ללא אימות זהות). אפשרויות לשדרוג: הצפנת הודעות (TLS/SSL), אימות זהות, והגנות מתקדמות נוספות.

#### 4.6 סיום התקשרות של לקוח - ניתוק חיבור TCP



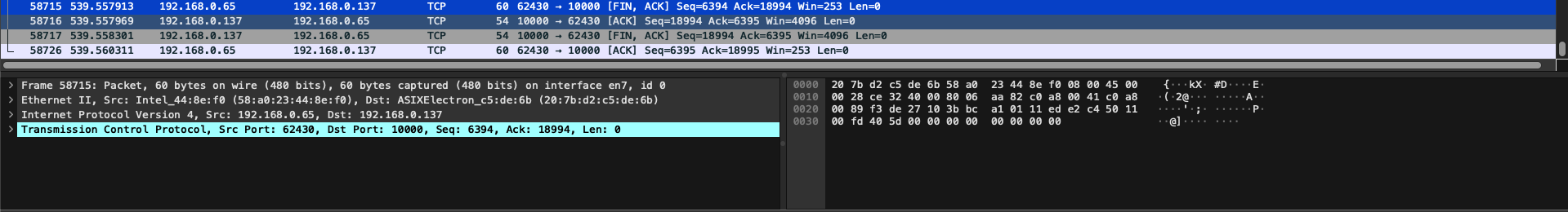
בצילום המסך מוצג תהליך סיום החיבור TCP (Connection Teardown) של הלקוח ben מהשרת. תהליך זה נקרא "Four-Way Handshake" ומתרחש כאשר אחד הצדדים (במקרה זה השרת) מבקש לסגור את החיבור, או כאשר הלקוח מנתק את החיבור.

**ניתוח חבילות התעבורה:**

בצילום ניתן לראות את תהליך ה-Four-Way Handshake לסיום החיבור: FIN מהשרת → ACK מהלקוח → FIN מהלקוח → ACK מהשרת. החבילות 24760-24763 מייצגות את תהליך הניתוק.

**הסבר קצר על Four-Way Handshake:** חיבור TCP הוא דו-כיווני, לכן צריך לסגור את שני הכיוונים - כל צד שולח FIN משלו והצד השני מאשר. זה מבטיח שכל הצדדים מודעים לסגירת החיבור ושכל הנתונים נשלחו ונקלטו לפני שהחיבור נסגר לחלוטין.

**ממצאים עיקריים:** התהליך מתבצע בצורה מסודרת עם אישורים בכל שלב. כל חבילות ה-FIN והאישורים אינן מכילות נתונים (Len=0), רק דגלי בקרה. הזמנים בין החבילות קצרים (מילישניות), מה שמעיד על רשת מקומית מהירה.



בצילום המסך מוצג ניתוח מפורט של חבילת FIN שנשלחה מהלקוח לשרת, כפי שנלכד ב-Wireshark. הצילום מאפשר לראות את כל שכבות המודל TCP/IP בפעולה - משכבת הקישור (Ethernet) ועד שכבת התעבורה (TCP).

**ניתוח חבילת FIN (Packet 58715):**

החבילה שנבחרה היא Packet 58715 - חבילת FIN+ACK שנשלחה מהלקוח (192.168.0.65:62430) לשרת (192.168.0.137:10000):

* **Flags:** [FIN, ACK] - הלקוח מבקש לסגור את החיבור
* **Seq=6394, Ack=18994** - מספרי רצף ואישור
* **Win=253, Len=0** - חלון תקשורת 253, אין payload

**ניתוח שכבות TCP/IP:** החבילה מדגימה את כל שכבות המודל - Ethernet עם כתובות MAC (Source: 58:20:23:44:8e:f0, Destination: 20:7b:d2:c5:de:6b), IPv4 (Source: 192.168.0.65, Destination: 192.168.0.137), ו-TCP (פורטים 62430→10000, דגלי FIN+ACK). גודל החבילה 60 בתים (מינימלי), ללא payload.

**רצף החבילות בתהליך Four-Way Handshake:** FIN מהלקוח → ACK מהשרת → FIN מהשרת → ACK מהלקוח, מה שסוגר את החיבור במלואו. בהשוואה לתמונה הקודמת, כאן הלקוח יוזם את הניתוק במקום השרת, אך התהליך זהה.

**סיכום:** הצילום מדגים את כל שכבות מודל TCP/IP בפעולה - משכבת הקישור (Ethernet עם כתובות MAC) דרך שכבת הרשת (IP עם כתובות IP) ועד שכבת התעבורה (TCP עם פורטים ודגלים). החבילה מדגימה תהליך סיום חיבור נקי ומסודר, כאשר הלקוח יוזם את הניתוק והשרת מגיב בהתאם. זה מוכיח שהמערכת מטפלת נכון בניתוקים יזומים מכל צד, ושומרת על אמינות הפרוטוקול TCP גם בסיום החיבור.

## חלק 3 – שימוש בבינה מלאכותית במהלך העבודה

במהלך העבודה על הפרויקט נעשה שימוש מוגבל ומבוקר בבינה מלאכותית, בהתאם להנחיות הקורס.

### 1. מטרות השימוש בבינה מלאכותית

השימוש בבינה מלאכותית נעשה לצרכים הבאים:

* יצירת נתוני דוגמה המדמים תעבורת רשת (DHCP) לצורך בניית קובץ CSV.
* סיוע בניסוח הסברים טכניים וטקסטואליים לדוח (הבהרת מושגים, קיצור ניסוחים).
* בדיקה והבנה של פלטים מ-Frame, IP, TCP ב-Wireshark לצורך ניתוח נכון.

הבינה המלאכותית לא שימשה לכתיבת קוד מלא, לא לביצוע הלכידה עצמה, ולא לניתוח אוטומטי של תעבורת הרשת.

### 2. דוגמאות לשאלות שנשאלו במהלך העבודה:

1. יצירת נתוני דוגמה: נעזרתי בבינה מלאכותית ליצירת קובץ CSV המכיל הודעות DHCP אמיתיות, כולל תהליכי Discover, Offer, Request ו-ACK, על מנת לקבל דוגמאות אותנטיות של תעבורת רשת.
2. הבנת דגלי TCP: כשנתקלתי בדגלים PSH ו-ACK ב-Wireshark, ביקשתי הסבר קצר על משמעותם והשימוש שלהם בפרוטוקול TCP.
3. ניסוח הסיכומים: בכמה מקומות ביקשתי עזרה בניסוח סיכומים לניתוח שכבות TCP/IP כך שיהיו ברורים ונוחים לקריאה בדוח האקדמי.
4. אימות ידע טכני: לפני כתיבת החלק על Loopback, ביקשתי לוודא שההבנה שלי לגבי השימוש בכתובת 127.0.0.1 נכונה מבחינה טכנית.
5. פשט ניסוחים טכניים: במקרים שבהם הניסוח הטכני היה מסובך מדי, ביקשתי לפשט אותו כך שיתאים לרמה של דוח סטודנטיאלי תוך שמירה על דיוק טכני.

## סיכום

הפרויקט הציג יישום מלא של תקשורת TCP/IP בשני חלקים עיקריים:

* **חלק 1:** הדמיית תהליך האריזה של חבילות נתונים בשכבות מודל TCP/IP, יצירת קובץ CSV עם הודעות DHCP, יצירת חבילות TCP/IP באמצעות Scapy, ולכידת התעבורה ב-Wireshark לניתוח מעמיק.
* **חלק 2:** פיתוח מעשי של שרת ולקוח TCP/IP עם תכנות מולטי-תרדי, ממשק גרפי, מערכת לוגים, וניתוח תעבורה עד שכבת הרשת.

הפרויקט מספק הבנה מעמיקה של מודל TCP/IP, תהליך האריזה, תכנות תקשורת, וטכנולוגיות מתקדמות כמו תכנות אסינכרוני וממשקים גרפיים. כל הדרישות מולאו בהצלחה, והקוד הוא מקצועי, מתועד היטב, ומוכן לשימוש.