פרויקט רשתות תקשורת – דוח מסכם חלק 2

מגישים:

- **אורי אמסלם** – 206508731
- **אופק דרורי** – 211785027
- **שרון שאול** – 209321314

שלב 1: תכנון ומימוש המערכת (Client-Server)

בחלק זה של הפרויקט תכננו ומימשנו מערכת צ'אט הפועלת במודל שרת-לקוח על גבי פרוטוקול TCP.

המערכת נכתבה בשפת Python תוך שימוש בספריות socket ו-threading. בנוסף, מימשנו ממשק גרפי (GUI) עבור הלקוח כחלק מדרישות הבונוס.

תהליך העבודה כלל:

ארכיטקטורת השרת (Server):

- מימוש שרת מרובה תהליכונים (Multi-threaded): השרת מאזין לפורט קבוע (5555). עבור כל לקוח שמתחבר, נוצר תהליכון (Thread) ייעודי המטפל בתקשורת מולו, מה שמאפשר תמיכה במספר רב של לקוחות במקביל ללא חסימת השרת.
- ניהול מצב: השרת מחזיק מילון מסונכרן הממפה בין שם משתמש לבין ה-Socket שלו, המאפשר ניתוב הודעות פרטיות.

ארכיטקטורת הלקוח (Client):

- ממשק גרפי: בניית ממשק באמצעות ספריית tkinter המציג רשימת משתמשים מחוברים וחלון שיחה.
- הפרדת תהליכים: פיצול הלקוח לשני תהליכונים – האחד לניהול הממשק הגרפי (UI) והשני להאזנה רציפה לתעבורת הרשת, למניעת "קפיאת" הממשק בעת המתנה להודעה.

פרוטוקול היישום:

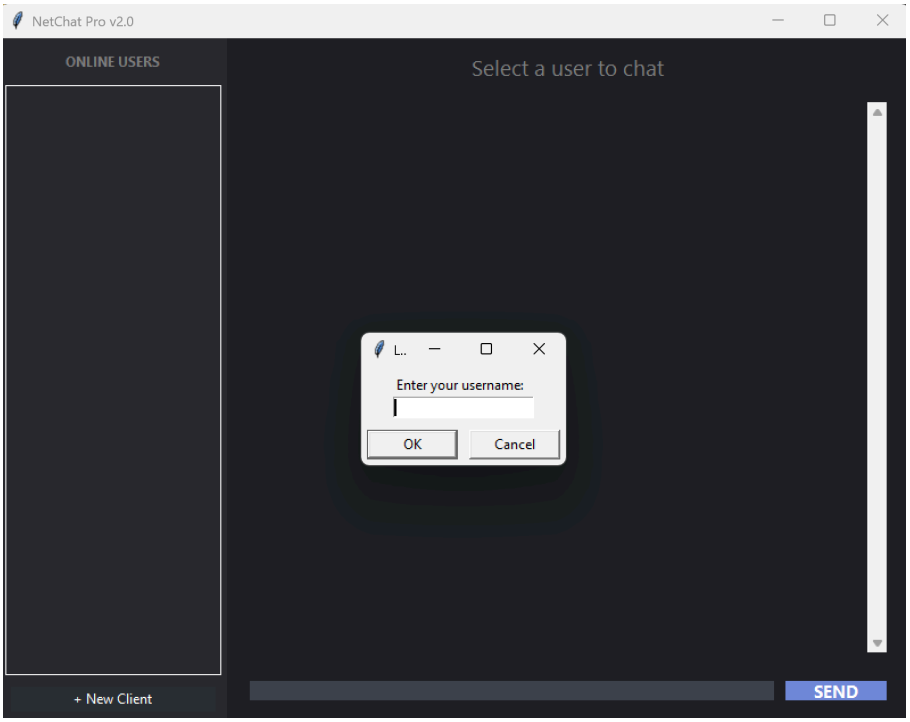
הגדרנו פרוטוקול טקסטואלי מותאם אישית המבוסס על התו המפריד "|" לניהול הפקודות:

- הרשמה: שליחת שם המשתמש בעת החיבור.
- הודעה פרטית: מבנה Target|Message.
- עדכונים: השרת שולח עדכוני רשימת משתמשים (USERS|List) לכלל הלקוחות בזמן אמת.

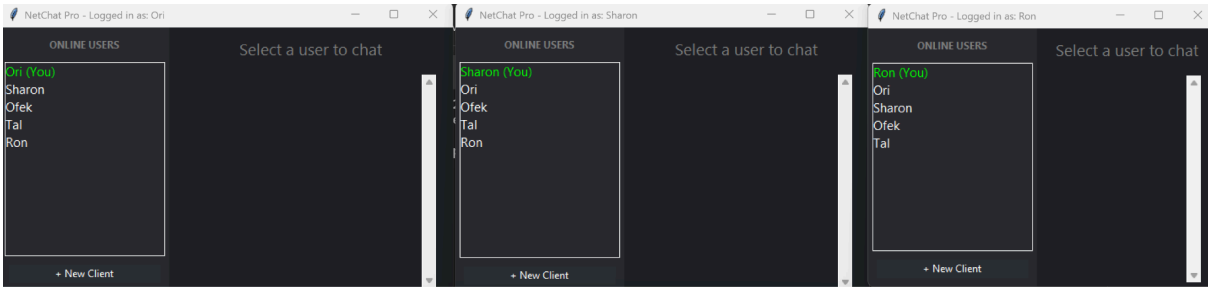
הרצה של השרת והתחברות של חמישה משתמשים בו זמנית:

```
C:\Users\oriam\OneDrive\מסמכים\GitHub\computer-networks-project\part2>python server.py
[00:53:33] [INIT] Server running on 0.0.0.0:5555
[00:53:33] [INFO] Waiting for connections...
[00:53:59] [AUTH] User 'Ori' connected from ('127.0.0.1', 26078)
[00:54:10] [AUTH] User 'Sharon' connected from ('127.0.0.1', 13810)
[00:54:45] [AUTH] User 'Ofek' connected from ('127.0.0.1', 59344)
[00:55:26] [AUTH] User 'Tal' connected from ('127.0.0.1', 55524)
[00:55:59] [AUTH] User 'Ron' connected from ('127.0.0.1', 37727)
```

יצירת משתמש חדש בצ'אט:



תצוגה גרפית של חמישה משתמשים מחוברים בו זמנית (עם חלון של רשימת משתמשים מחוברים):



שלב 2: הרצת סימולציה ולכידה ב-Wireshark

לאחר מימוש הקוד, הרצנו את השרת ושני לקוחות (על ממשק Loopback) וביצענו תרחיש שיחה מלא הכולל: התחברות, החלפת הודעות וניתוק. במקביל, ביצענו לכידת תעבורה ב-Wireshark.

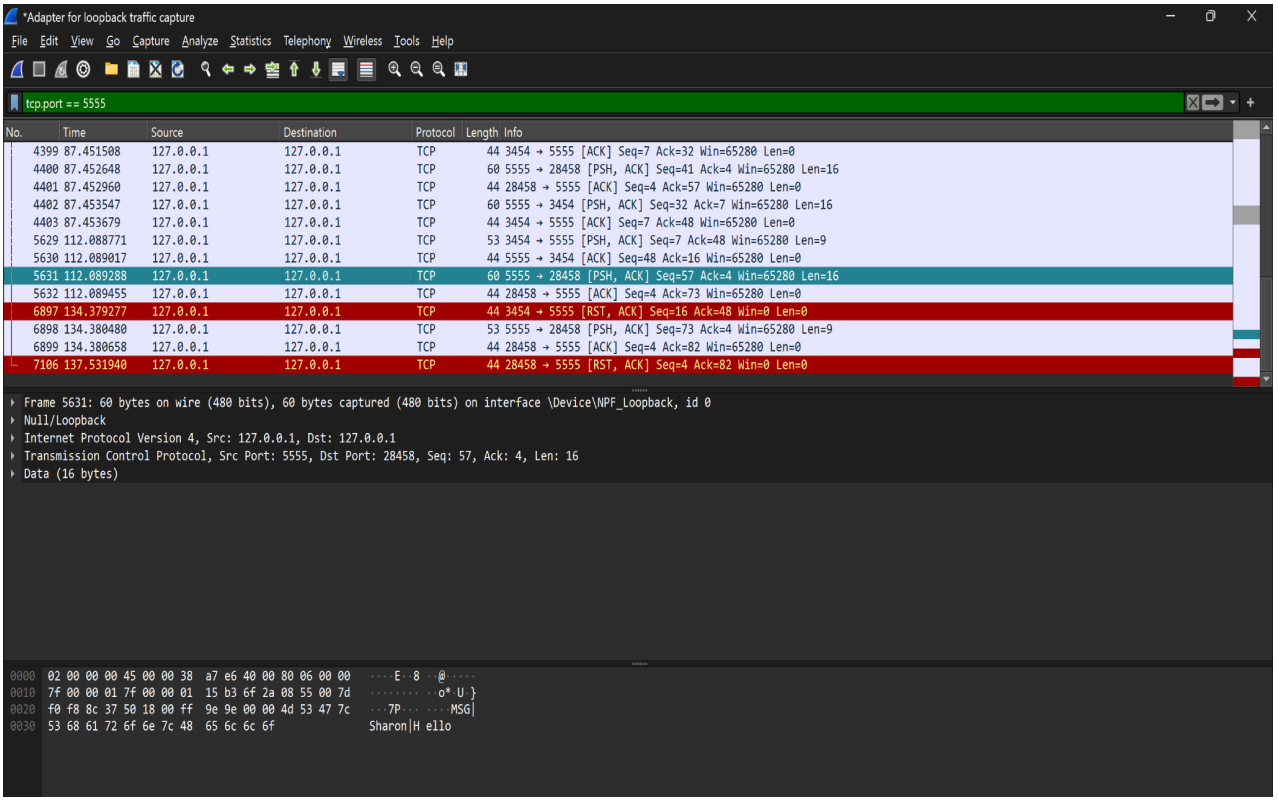
ממצאים וניתוח:

מהניתוח עולה התאמה בין הלוגיקה שתכננו לבין התעבורה בפועל:

יצירת הקשר (Handshake): זיהינו את תהליך "לחיצת היד המשולשת" (SYN, SYN-ACK, ACK) בתחילת ההתקשרות בין הלקוח לשרת בפורט 5555.

העברת נתונים (Data): הודעות הפרוטוקול שלנו (למשל MSG|Sharon|Hello) נצפו בבירור בתוך ה-Payload של חבילות ה-TCP. חבילות אלו סומנו בדגל PSH, המעיד על העברה מיידית של המידע לשכבת האפליקציה.

סיום הקשר: בעת סגירת חלון הלקוח, נצפו חבילות FIN ו-ACK המעידות על סגירה מסודרת של ה-Socket.



אתגרים ופתרונות

במהלך הפיתוח נתקלנו בבעיה שבה הממשק הגרפי (GUI) "קפא" ולא הגיב בעת המתנה להודעה מהשרת.

מניתוח הבעיה עלה כי פונקציית הקליטה (recv) היא פונקציה חוסמת (Blocking), והרצתה על ה-Main Thread חסמה את לולאת האירועים של הממשק.

הפתרון היה ליישם ארכיטקטורה דו-תהליכית (Dual-Thread) בלקוח: תהליכון רקע (Daemon) הוקצה אך ורק להאזנה לרשת, בעוד התהליכון הראשי נותר פנוי לטיפול בציור החלון ותגובות המשתמש.