

דוח מסכם: פרויקט תקשורת מחשבים - חלק 2

מגישות: נטע גולזאד 213487200, עדן זרביאן 212055024

https://github.com/edenzarbian/networking_project

הסבר כללי על המערכת ומבנה הקוד:

המערכת שפותחה היא יישום צ'אט מבוסס רשת, המאפשר תקשורת ישירה בין משתמשים בזמן אמת. היישום בנוי בארכיטקטורת Client - Server, שבה שרת מרכזי מנהל את החיבורים ומתווך את העברת ההודעות בין הלקוחות.

התקשורת בין רכיבי המערכת מתבצעת באמצעות פרוטוקול TCP, המבטיח העברת נתונים אמינה, שמירה על סדר ההודעות וטיפול בשגיאות תקשורת. כל לקוח מתחבר לשרת, מזדהה באמצעות שם משתמש ייחודי, ויכול לנהל שיחות פרטיות עם משתמשים אחרים המחוברים למערכת.

השרת אינו שומר מידע מתמשך, אלא משמש כמתווך להעברת הודעות בלבד. המערכת תומכת במספר לקוחות בו-זמנית ומדמה עקרונות בסיסיים של מערכות מבזרות ויישומי מסרים מיידיים.

רכיבי המערכת:

א. צד שרת - server.py

השרת מהווה את הרכיב המרכזי במערכת ואחראי על ניהול החיבורים והעברת ההודעות בין הלקוחות.

- השרת יוצר Socket מבוסס TCP, מאזין על הכתובת 127.0.0.1 ובפורט 65432, ומקבל חיבורים נכנסים מלקוחות.
- לכל לקוח שמתחבר נוצר Thread ייעודי המטפל בקשר מולו באמצעות הפונקציה `handle_client`, כך השרת מסוגל לשרת מספר לקוחות במקביל.
- בכל התחברות או ניתוק השרת שולח לכל הלקוחות הודעת מערכת בפורמט: `USER_LIST_UPDATE:user1,user2...` לצורך עדכון רשימת המשתמשים המחוברים.

ניהול משתמשים וסנכרון (Lock):

- השרת שומר מילון בשם `clients` הממפה בין שם משתמש לחיבור ה-Socket שלו. בעת התחברות המשתמש נוסף למילון, ובעת ניתוק הוא מוסר ממנו.

פרוטוקול האפליקציה:

- התחברות - שליחת שם משתמש מיד לאחר החיבור
- שליחת הודעה - target:message
- קבלת הודעה - sender:message
- שגיאה - SEND_FAILED:User <name> is no longer online

ב. צד לקוח - client.py

הלקוח הוא יישום קצה הכולל ממשק גרפי (GUI) המאפשר למשתמש להתחבר לשרת, לבחור משתמש יעד ולשלוח ולקבל הודעות בזמן אמת. הממשק ממומש באמצעות tkinter ומציג רשימת משתמשים מחוברים וחלון שיחה.

כדי למנוע קיפאון של הממשק בזמן קבלת הודעות, קליטת הנתונים מהשרת מתבצעת ב־Thread נפרד. כך ה-GUI נשאר פעיל ומתעדכן באופן רציף במהלך השיחה.

הוראות התקנה והרצה:

דרישות מקדימות

- התקנת Python 3.x.
- קבצי הפרויקט נמצאים באותה תיקייה: client.py, server.py.

הפעלת השרת

1. פתח Terminal / CMD בתיקיית הפרויקט
2. הרץ python server.py
3. ודא שמתקבלת הודעה: Server is Up and listening

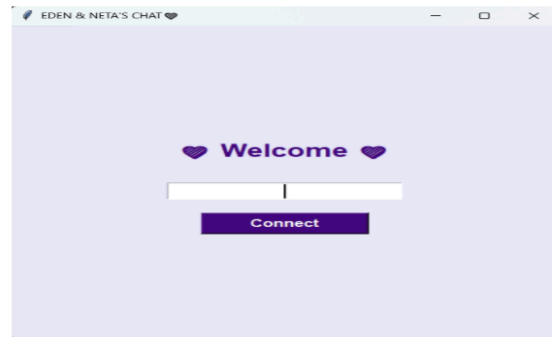
הפעלת הלקוחות

1. פתח חלון Terminal נוסף (ולכל לקוח נוסף פתחו חלון נוסף)
2. הרץ python client.py
3. הזן שם משתמש ייחודי ולחצו Connect.
4. חזור על הפעולה עבור לקוחות נוספים עם שמות שונים כדי לבצע שיחה בין משתמשים

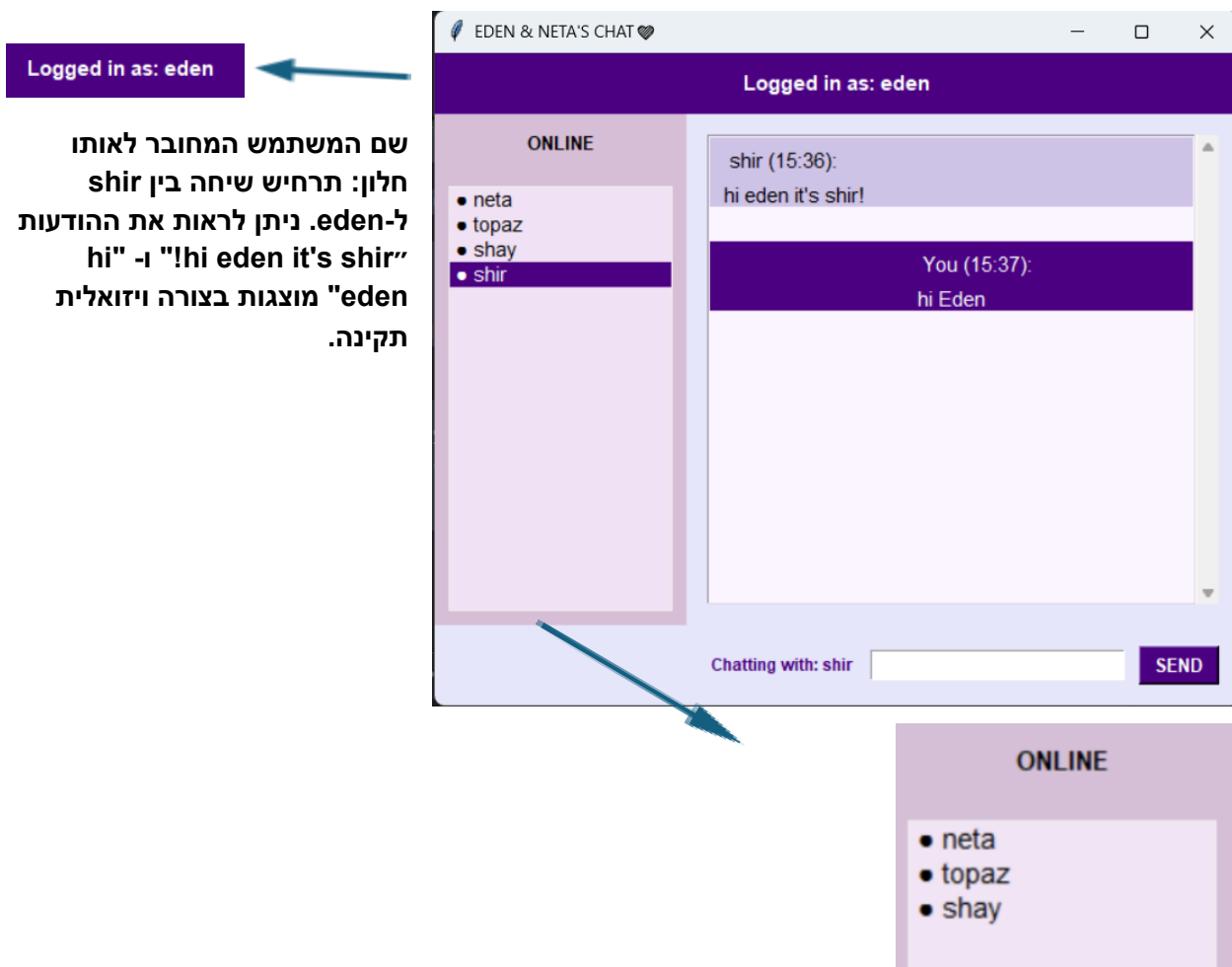
המערכת רצה מקומית על 127.0.0.1 בפורט 65432.

דוגמאות קלט ופלט (GUI):

מסך הפתיחה של היישום, שבו המשתמש מזין את שם המשתמש שלו ולוחץ על כפתור Connect. לאחר ההתחברות, שם המשתמש נשלח לשרת לצורך זיהוי, והמשתמש מועבר למסך הצ'אט הראשי.



להלן תצוגת המערכת בזמן פעולה. ניתן לראות את רשימת אנשי הקשר בצד שמאל, ואת חלון הצ'אט עם ההודעות (סגול כהה עבורי, סגול בהיר עבור הצד השני).



שם המשתמש המחובר לאותו חלון: תרחיש שיחה בין shir ל-eden. ניתן לראות את ההודעות "hi eden it's shir" ו-"hi" "eden" מוצגות בצורה ויזואלית תקינה.

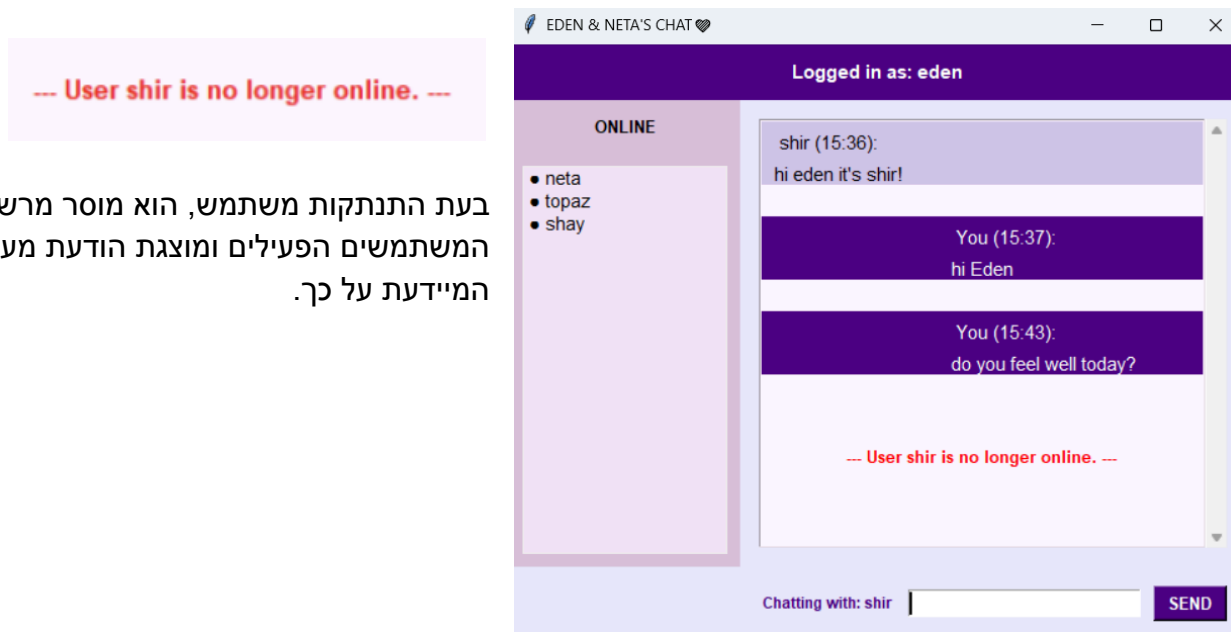
חלון המציג את כל המשתמשים המחוברים כעת לשרת בזמן אמת, הרשימה מתעדכנת בכל התחברות או ניתוק של משתמש בהתאם להודעות עדכון שמתקבלות מהשרת. בחירת משתמש מהרשימה מגדירה אותו כיעד לשיחה.

תהליך שליחת ההודעה:

1. בחירת יעד - המשתמש בוחר איש קשר מרשימת המשתמשים המחוברים בצד שמאל של הממשק.
2. הקלדת ההודעה - המשתמש מקליד את ההודעה בשדה הטקסט בתחתית חלון הצ'אט.
3. שליחה - בעת לחיצה על כפתור השליחה, הלקוח יוצר מחרוזת בפורמט `target:message` ושולח אותה לשרת באמצעות חיבור ה-TCP.
4. עיבוד בשרת - השרת מקבל את ההודעה, מזהה את יעד ההודעה, ובודק האם המשתמש היעד מחובר.
5. העברת ההודעה - אם היעד מחובר, השרת מעביר אליו את ההודעה בפורמט `sender:message`.
6. הצגה בממשק - ההודעה מוצגת מיידית בחלון הצ'אט:
 - בצד השולח – סגול כהה
 - בצד המקבל – סגול בהיר

בכך מתבצעת שליחת הודעה בזמן אמת בין שני משתמשים דרך השרת.

בעת התנתקות משתמש, הוא מוסר מרשימת המשתמשים הפעילים ומוצגת הודעת מערכת המיידעת על כך.



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Server is Up and listening
[SERVER] eden connected.
[SERVER] neta connected.
[SERVER] topaz connected.
[SERVER] shay connected.
[SERVER] shir connected.
[SERVER] shir disconnected.
```

מה קורה בצד השרת?

השרת מאזין לחיבורים נכנסים ומקבל מכל לקוח את שם המשתמש שלו. עם כל התחברות, השרת מוסיף את המשתמש למילון המשתמשים המחוברים ומדפיס הודעת התחברות בחלון ה-Terminal. כאשר לקוח מתנתק (באופן יזום או לא צפוי), השרת מזהה את הניתוק, מסיר את המשתמש מהמילון ומדפיס הודעת התנתקות.

מה קורה בצד הלקוח?

בצד הלקוח, כל התחברות או ניתוק של משתמש מתבטאים בעדכון הממשק הגרפי. כאשר משתמש יעד מתנתק במהלך שיחה, הלקוח מקבל מהשרת הודעת מערכת ומציג הודעה מתאימה בחלון הצ'אט, המיידעת כי המשתמש אינו מחובר עוד.

ניתוח זרימת תעבורה: מערכת צ'אט מבוססת TCP

1. מבוא ותשתית הרשת

הפרויקט מבוסס על תקשורת בכתובת ה-Loopback המקומית (127.0.0.1), המאפשרת תקשורת בין תהליכים על אותו מחשב. התקשורת מתבצעת בשכבת התעבורה באמצעות פרוטוקול TCP, המבטיח אמינות והעברת נתונים לפי סדר.

22	55.354046	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432 [ACK] Seq=1 Ack=1 Win=65280 Len=0
23	55.354088	127.0.0.1	127.0.0.1	TCP	49	61110 → 65432 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=5
24	55.354095	127.0.0.1	127.0.0.1	TCP	44	65432 → 61110 [ACK] Seq=1 Ack=6 Win=65280 Len=0
25	55.355173	127.0.0.1	127.0.0.1	TCP	71	65432 → 61107 [PSH, ACK] Seq=22 Ack=5 Win=65280 Len=27
26	55.355217	127.0.0.1	127.0.0.1	TCP	44	61107 → 65432 [ACK] Seq=5 Ack=49 Win=65280 Len=0
27	55.355239	127.0.0.1	127.0.0.1	TCP	71	65432 → 61110 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=27
28	55.355259	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432 [ACK] Seq=6 Ack=28 Win=65280 Len=0
29	63.408416	127.0.0.1	127.0.0.1	TCP	56	61110 → 65432 [PSH, ACK] Seq=6 Ack=28 Win=65280 Len=12[Malformed Packet]
30	63.408446	127.0.0.1	127.0.0.1	TCP	44	65432 → 61110 [ACK] Seq=28 Ack=18 Win=65280 Len=0
31	63.408513	127.0.0.1	127.0.0.1	TCP	57	65432 → 61107 [PSH, ACK] Seq=49 Ack=5 Win=65280 Len=13
32	63.408547	127.0.0.1	127.0.0.1	TCP	44	61107 → 65432 [ACK] Seq=5 Ack=62 Win=65280 Len=0
33	69.050817	127.0.0.1	127.0.0.1	TCP	44	61107 → 65432 [RST, ACK] Seq=5 Ack=62 Win=0 Len=0
34	69.051337	127.0.0.1	127.0.0.1	TCP	66	65432 → 61110 [PSH, ACK] Seq=28 Ack=18 Win=65280 Len=22
35	69.051378	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432 [ACK] Seq=18 Ack=50 Win=65280 Len=0
36	80.063388	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432 [RST, ACK] Seq=18 Ack=50 Win=0 Len=0

0000	02 00 00 00 45 00 00 43 14 39 40 00 80 06 00 00E..C.9@.....
0010	7f 00 00 01 7f 00 00 01 ff 98 ee b3 1a 9f 62 3db=
0020	6a 93 88 33 50 18 00 ff 61 7c 00 00 55 53 45 52	j..3P...a .USER
0030	5f 4c 49 53 54 5f 55 50 44 41 54 45 3a 65 64 65	_LIST_UP DATE:ede
0040	6e 2c 74 6f 70 61 7a	n,topaz

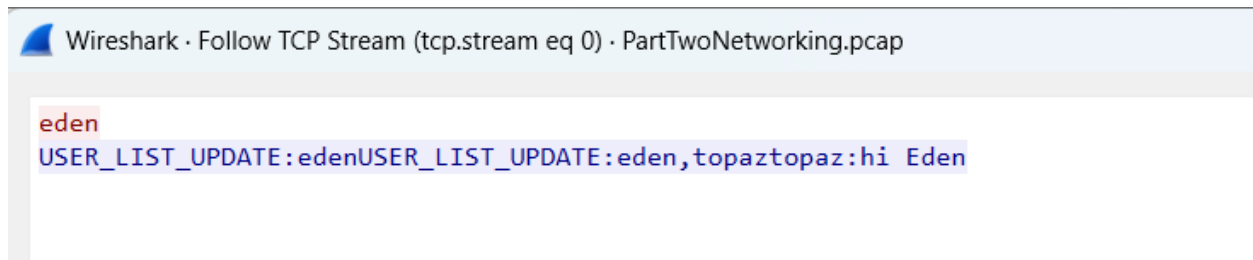
- בתמונה זו ניתן לראות תקשורת חיה בין לקוח לשרת בכתובת המקומית (127.0.0.1).

2. הקמת חיבור ושימוש תקין

כאשר השרת פעיל ומאזין בפורט, ניתן לראות את שלבי החיבור התקינים:

- כל הודעה שנשלחת זוכה לאישור קבלה (ACK) מהצד השני.
- ניתן לראות שה-Sequence Number גדל בהתאם לאורך ההודעה, מה שמוכיח שהמידע הגיע בשלמותו ובסדר הנכון.

3. ניתוח המידע באמצעות פעולת Follow TCP Stream



- **עדכוני מערכת:** ניתן לראות הודעות פרוטוקול פנימיות כמו USER_LIST_UPDATE:eden, המעידות על כך שהשרת מעדכן את הלקוח על הצטרפות משתמש חדשה.
- **תוכן הצ'אט:** רואים העברת טקסט חופשי בין המשתמשים למשל: topaztopaz:hi Eden.

4. קטיעת החיבור:

במהלך הניסוי נדגמו מצבים של "ניתוקים אלימים", המזוהים ב-Wireshark בצבעים אדום ושחור:

- **זיהוי הבעיה:** מופיעה חבילה עם דגל [RST, ACK]. בניגוד לסגירה רגילה, כאן אחד הצדדים שלח פקודת איפוס (Reset).
- **משמעות:** החיבור נקטע בצורה אלימה. זה יכול לקרות בגלל סגירה פתאומית של תוכנת הצ'אט, קריסה של השרת, או ניסיון לשלוח מידע לפורט שכבר נסגר.

32	63.408547	127.0.0.1	127.0.0.1	TCP	44	61107 → 65432	[ACK] Seq=5 Ack=62 Win=65280 Len=0
33	69.050817	127.0.0.1	127.0.0.1	TCP	44	61107 → 65432	[RST, ACK] Seq=5 Ack=62 Win=0 Len=0
34	69.051337	127.0.0.1	127.0.0.1	TCP	66	65432 → 61110	[PSH, ACK] Seq=28 Ack=18 Win=65280 Len=2
35	69.051378	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432	[ACK] Seq=18 Ack=50 Win=65280 Len=0
36	80.063388	127.0.0.1	127.0.0.1	TCP	44	61110 → 65432	[RST, ACK] Seq=18 Ack=50 Win=0 Len=0

תיאור שימוש בבינה מלאכותית (AI)

במהלך פיתוח הפרויקט נעשה שימוש במודל שפה (LLM) כעזר תומך בתהליך הפיתוח, התכנון והניתוח, ולא כחלק פונקציונלי ממערכת הצ'אט עצמה.

מטרות השימוש ב-AI:

- סיוע בתכנון ועיצוב ממשק המשתמש (GUI) באמצעות tkinter, כולל מבנה חלון הצ'אט, חלוקת אזורים ועיצוב בועות הודעה בהשראת יישומי מסרים קיימים.
- סיוע בהבנת התנהגות תעבורת TCP שנצפתה ב-Wireshark.
- סיוע בתהליך דיבוג (Debugging), ובפרט בזיהוי בעיות קיפאון בממשק הגרפי ופתרון באמצעות שימוש ב-Thread נפרד לפונקציית קבלת ההודעות (receive).

דוגמאות לפרומפטים ששימשו אותנו:

- "איך ניתן לנהל מספר שיחות פרטיות במקביל ביישום צ'אט מבוסס TCP?"
- "כיצד ניתן להבחין ב-Wireshark בין חבילות שליחת הודעה לבין חבילות אישור (ACK)?"
- "כיצד ניתן לזהות ב-Wireshark ניתוק של לקוח משרת TCP ומהם הסימנים לכך בתעבורה?"
- "איך לבדוק האם הבעיה היא בצד הלקוח או בצד השרת כאשר אין תגובה בממשק?"