

דוח מסכם: פרויקט תקשורת מחשבים - חלק 2

מגישות: נטע גולזאדי 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 בניו בארכיטקטורת Client - Server, שבה שירות מרכזי מנהל את החיבורים ומתווך את העברת ההודעות בין הלקוחות.

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

השרת אינו שומר מידע מתמשך, אלא משמש כמתווך להעברת הודעות בלבד. המערכת תומכת במספר לקוחות בו-זמנית ומדמה עקרונות בסיסיים של מערכות מבוזרות 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, מזמן על הכתובת 127.0.0.1 ופורט 65432, ומקבל חיבורים נוספים מלקוחות.
- כל לקוח שמתחבר נוצר Thread "יעודי המטפל בקשר מולו באמצעות הפונקציה handle_client", כך השרת מסוגל לשרת מספר לקוחות בו-זמנית.
- בכל התחרבות או ניתוק השרת שולח לכל הלקוחות הודעה מערכת בפורמט: USER_LIST_UPDATE:user1,user2...

:ניהול משתמשים וסנכרון (Lock)

- השרת שומר מילון בשם clients הממפה בין שם משתמש לחיבור ה-socket שלו. בעת התחרבות 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
- SEND_FAILED:User <name> is no longer online - שגיאה

ב. צד לקוח - client.py

הלקוח הוא יישום קצה הכלול מממשק גרפי (GUI) המאפשר למשתמש להתחבר לשרת, לבחור משתמש יעד ושלוח ו לקבל הודעה בזמן אמת. הממשק ממומש באמצעות `tkinter` ומציג רשות משתמשים מחוברים וחילון שיחה.

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

הוראות התקינה והרכבה:

דרישות מקדים:

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

הפעלת השרת

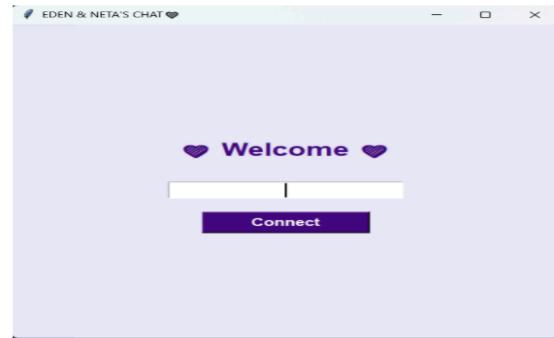
1. פתח Terminal / CMD בתיקיית הפרויקט
2. הרץ `python server.py`
3. ודה שמתתקבלת הודעה: Server is Up and listening

הפעלת לקוחות

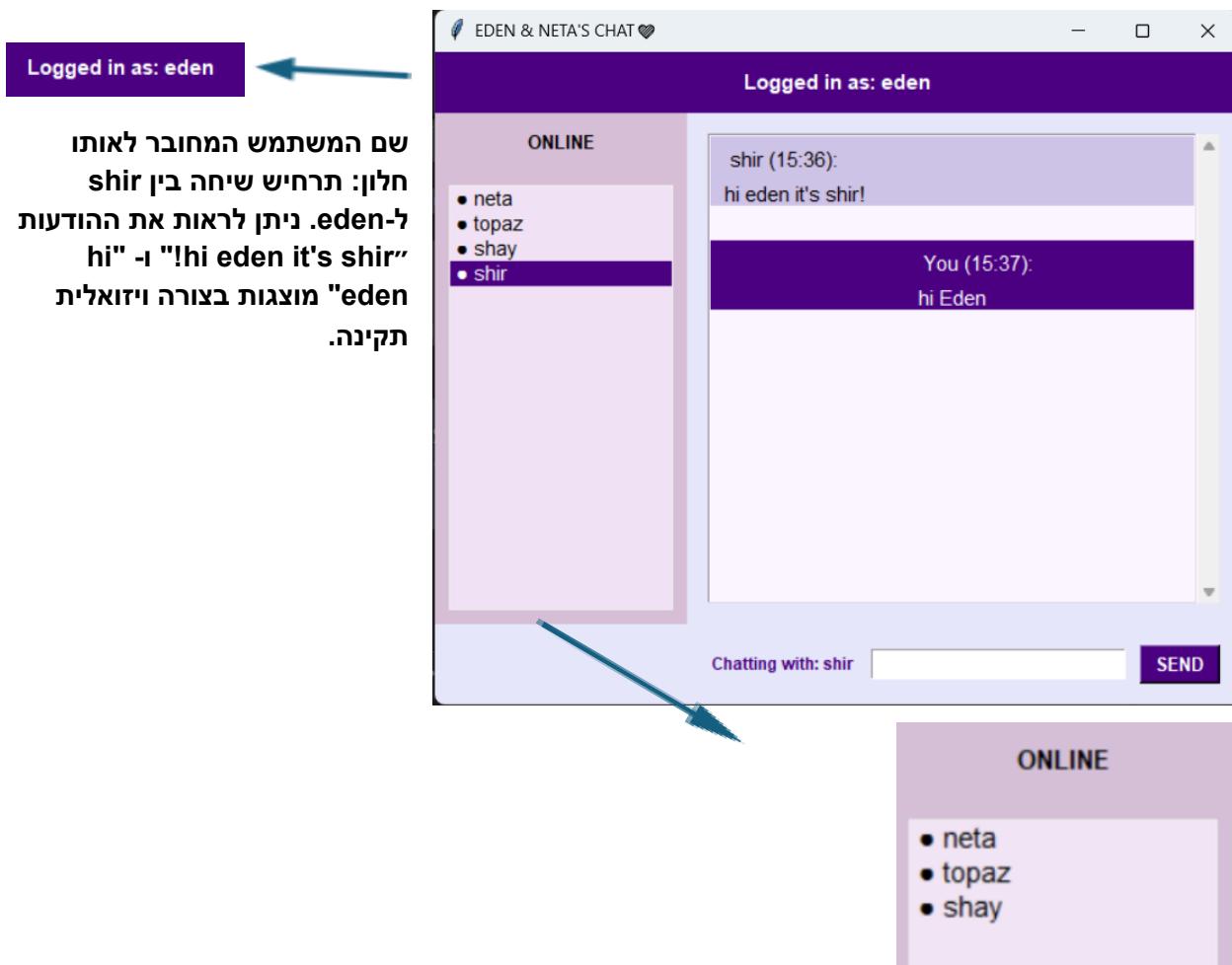
1. פתח חילון Terminal נוסף (ולכל לקוח נוסףفتحו חילון נוסף)
 2. הרץ `python client.py`
 3. הzin שם משתמש ייחודי ולהצט `Connect`.
 4. חזר על הפעולה עבור לקוחות נוספים עםשמות שונים כדי לבצע שיחה בין משתמשים
- המערכת רצה מקומית על 127.0.0.1 בפורט 65432.

דוגמאות קלט ופלט (GUI):

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



להלן תצוגת המערכת בזמן פעולה. ניתן לראות את רשימת אנשי הקשר הצד שמאל, ואת חלון הציג עם ההודעות (סגול כהה עליון, סגול בהיר עבור הצד השני).



Logged in as: eden

שם המשתמש המחבר לאותו חלון: **shir**. ניתן לראות את ההודעות ל-eden. ניתן לראות את ההודעות **"hi eden it's shir"** ו- **"hi"** מוצגות בצהורה ויזואלית תקינה.

shir (15:36):
hi eden it's shir!

You (15:37):
hi Eden

Chatting with: shir

SEND

ONLINE

- neta
- topaz
- shay
- shir**

ONLINE

- neta
- topaz
- shay

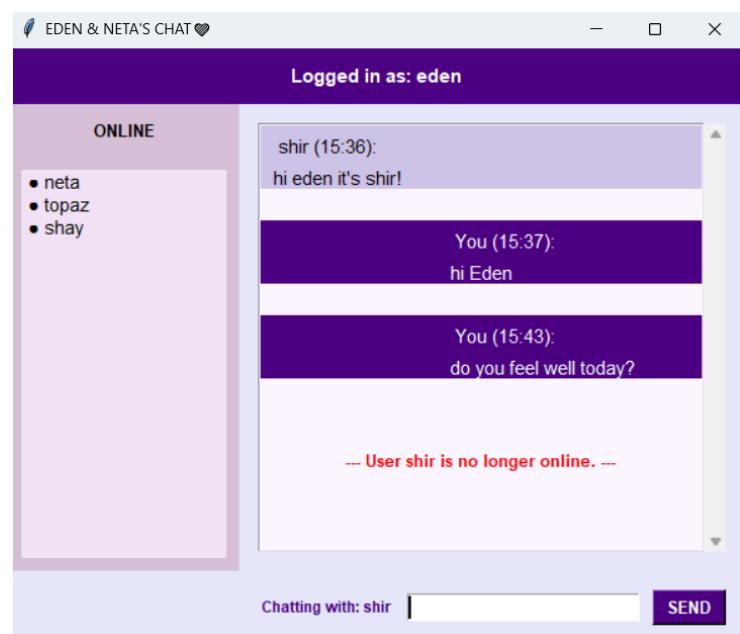
חלון המציג את כל המשתמשים המחברים כתע לשרת בזמן אמיתי, הרשימה מתעדכנת בכל התחברות או ניתוק של משתמש בהתאם להודעות עדכון שמקבלות מהשרת. בחירת משתמש מהרשימה מגדרה אותו כיעד לשיחת.

תהליך שליחת ההודעה:

1. בחירת יעד - המשתמש בוחר איש קשר מרשימה המשתמשים המוחברים מצד שמאל של המסך.
2. הקלדת הודעה - המשתמש מקליד את הודעה בשדה הטקסט בתחתית חלון הצ'אט.
3. שליחה - בעט ללחיצה על כפתור השליחה, הלקוח יוצר מחרוזת בפורמט target:message ושולח אותה לשרת באמצעות חיבור ה-TCP.
4. עיבוד בשרת - השרת מקבל את הודעה, מזיהה את יעד הודעה, ובודק האם המשתמש היעד מוחבר.
5. העברת הודעה - אם היעד מוחבר, השרת מעביר אליו את הודעה בפורמט sender:message
6. הצגה במכשיר - הודעה מוצגת מיידית בחלון הצ'אט:
 - בצד השולח – סגול כהה
 - בצד מקבל – סגול בהיר

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

בעת התנטקנות משתמש, הוא מסור מרשימה המשתמשים הפעילים ומוצגת הודעה מערכת המידע על כך.



The screenshot shows a terminal window with four tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected and underlined. The window displays a log message from a server. The message consists of several lines of text in white font on a black background, indicating connections and disconnections from various clients named eden, neta, topaz, shay, and shir.

```
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 מודפס למסך נכנים ומתקבל מכל לקוח את שם המשתמש שלו. עם כל התוכנות, השרת מוסיף את המשתמש למילון המשתמשים המוחברים ומדפיס הודעה התוכנות בחילון ה- Terminal. כאשר לקוח מתנתק (באופן יזום או לא צפוי), השרת מזהה את הניתוק, מסיר את המשתמש מהamilon ומדפיס הודעה התונתקות.

מה קורה בצד הלוקוט?

בצד הלוקוט, כל התוכנות או ניתוק של משתמש מرتبطים בעדכון הממשק הגרפי. כאשר משתמש יעד מתנתק במהלך שיחה, הלוקוט מקבל מהשרת הודעה מערכת ומציג הודעה מתאימה בחילון הcz'אט, המיידעת כי המשתמש אינו מחובר עוד.

ניתוח זרימת תעבורה: מערכת צ'אט מבוססת TCP

1. מבוא ותשתיות הרשת

הפרויקט מוביל עלי תקשורת בכתובת-loopback המקומית (127.0.0.1), המאפשרת תקשורת בין הליינס על אותו מחשב. התקשרות מתבצעת בשכבה 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=5 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

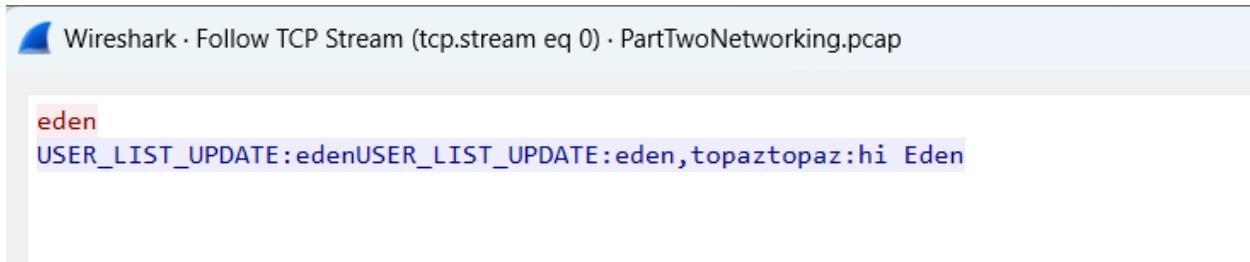
- בתמונה זו ניתן לראות תקשורת חיה בין ללקוח לשרת בכתובת המקומית (127.0.0.1).

2. הקמת חיבור ושימוש תקין

כasher השרת פועל ומАЗין 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 שומוכח שהມידע הגיע בשלומו ובסדר הנכון.

3. ניתוח המידע באמצעות פעולה TCP Stream



Wireshark · Follow TCP Stream (tcp.stream eq 0) · PartTwoNetworking.pcap

eden

USER_LIST_UPDATE:edenUSER_LIST_UPDATE:eden,topaztopaz:hi Eden

- עדכוני מערכת: ניתן לראות הודעות פרוטוקול פנימיות כמו eden:USER_LIST_UPDATE:eden המעידות על כך שהשרת מעדכן את הלוקה על הלקוח משתמש מחדש.
- תוקן היצ'אט: רואים העברת טקסט חופשי בין המשתמשים למשל: hi Eden .topaztopaz:hi Eden

4. קטיעת החיבור:

במהלך הניסוי נדגמו מצבים של "ניתקים אלימים", המזוהים ב-Wireshark כצבעים אדום ושחור:

- זיהוי הבעיה: מופיעה חבילה עם דגל [RST, ACK]. בኒוגוד לסגירה רגילה, כאן אחד הצדדים שלח פקודת איפוס (Reset).
- משמעות: החיבור נקטע بصورة אלימה. זה יכול לקרות בגין סגירה פתאומית של תוכנת היצ'אט, קירישה 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 במבנה מלאכותית (AI)

במהלך פיתוח הפרויקט נעשה שימוש במודל שפה (LLM) כדי תומך בתהיליך הפיתוח, התכוון והניתוח, ולא כחלק פונקציוני ממערכת ה策'ט עצמה.

מטרות השימוש ב-AI:

- סיוע בתכנון ועיצוב מוצר המשמש (GUI) באמצעות tkinter, כולל מבנה חלון策'ט, חלוקת אזורים ועיצוב בוועות הودעה בהשראת יישומי מסרים קיימים.
- סיוע בהבנת התנהלות תעבורת TCP שנצפתה ב-Wireshark.
- סיוע בתהיליך דיבוג (Debugging), ובפרט בזיהוי בעיות קיפאון במכשיר הגרפי ופתרון באמצעות שימוש ב-thread receive.

דוגמאות לrogramפטיים ששימשו אותנו:

- "איך ניתן לנצל מספר שיחות פרטיות במקביל ביחסו策'ט TCP?"
- "كيف يمكن للتحقق في Wireshark بين خيارات إرسال الرسالة وبين خيارات التأكيد (ACK)؟"
- "كيف يمكن التعرف علىWireshark على أنه服务 TCP ومتى هي الماركيمات التي تأتي في الرسالة؟"
- "איך לבדוק האם הבעיה היא בצד הלוקוח או הצד השרת כאשר אין תגובה במכשיר?"