איתי קרואני 311504914
טל מלחי 208278556

פרויקט סיום תקשורת

פרויקט זה מייצג מערכת מסרים מיידית המבוססת על תקשורת.
המערכת מורכבת מצד-שרת וצד לקוח, כאשר צד השרת יושב על השרת, וצד הלקוח מכיל ממשק גרפי לנוחות המשתמש. עבור כל לקוח חדש שמתחבר לצ'ט (לשרת) ע"י חיבור TCP , ניתנת האופציה לבצע מספר פעולות.

צד השרת :

ממומש ע"י סוקט המאזין לחיבורים נכנסים. עבור כל חיבור חדש, השרת פותח Thread חדש אשר ינהל את החיבור מול המשתמש, כך שבמידה ותהיה בעיה בחיבור עם משתמש מסוים לא כל הקוד של השרת ייתקע, אלא רק אותו טרד. בדרך זו נוכל לטפל ביעילות בשגיאות, ובנוסף תתאפשר הורדה יעילה יותר של קבצים ע"י לקוחות שונים, כולל הורדה במקביל של קבצים שונים מלקוחות שונים.

באשר לניהול החיבורים, השרת מריץ לולאה אשר מחכה לחיבורים נכנסים ולבקשות מהלקוח.
כאשר משתמש חדש מתחבר לשרת, הוא מוסיף אותו למילון המשתמשים הקיימים, כאשר המפתח הוא שם הלקוח והערך הוא הסוקט שלו. לשרת וללקוח קיימת שפה משותפת בה הם יכולים "לתקשר ביניהם" כדי שהשרת יידע לטפל בכל לקוח בהתאם לבקשתו, ולהחזיר לו את התשובה בהתאם.
הטיפולים האפשריים עבור בקשת הלקוח:

1. LOGIN – נוסיף את הלקוח לרשימת הלקוחות המחוברים לשרת (active_cllients). במידה והחיבור הצליח, נחזיר ללקוח הודעה כי אכן החיבור צלח. אם הלקוח כבר מחובר לשרת, נחזיר הודעה שמשתמש זה כבר מחובר.
2. LOGOUT – נסגור את הסוקט של הלקוח, נסיר אותו מרשימת המשתמשים המחוברים ונשלח הודעה בצ'ט שאותו לקוח התנתק מהצ'ט.
3. SHOWFILES- מחזיר ללקוח את כל הקבצים שנמצאים בשרת.
4. SHOWUSERS- מחזיר ללקוח את כל המשתמשים המחוברים לצ'ט, בהתאם ל active_cllients
5. MSG- מציג את ההודעה אותה שלח הלקוח לכל המשתמשים המחוברים לצ'ט, באמצעות פונקציית broadcast, אשר שולחת הודעה לכל המחוברים בצ'ט בהתאם ל active_cllients
6. PMSG– מציג את ההודעה אותה שלח הלקוח למשתמש מסויים
7. DOWNLOAD- מוריד את הקובץ אותו ביקש הלקוח. הורדת הקובץ מתבצעת בצורה אמינה מעל UDP בטרד נפרד, לכן השרת יוכל לטפל בבקשות נוספות אחרות במקביל.

צד הלקוח:

מנוהל על ידי ממשק גרפי. בחיבור הראשוני נפתח סוקט מצד הלקוח אשר מתחבר לשרת על ידי חיבור TCP. חיבור זה מוצג למשתמש באמצעות כפתור ה start בממשק הגרפי. כאשר צלח החיבור, נדרש הלקוח להכניס שם, וכתובת וללחוץ על כפתור login בממשק הגרפי. כשהלקוח הצליח להתחבר לצ'ט (online), נפתח Thread חדש אשר יאזין לתגובות השרת, בהתאם לבקשת הלקוח. באפשרות כל לקוח לבצע מספר פעולות, אשר מיוצגות בממשק גרפי באמצעות כפתורים: להתחבר ולהתנתק מהצ'ט, לקבל את רשימת המחוברים לצ'ט, לשלוח הודעה לכל הלקוחות שמחוברים, לשלוח הודעה ללקוח מסוים, לקבל את רשימת הקבצים שיש בשרת, להוריד קובץ, לעצור את ההורדה באמצע ולהמשיך. צד זה מריץ לולאה שמחכה לפעולות הלקוח. כל בקשה של הלקוחות תישלח לשרת, ואת תגובתה נקבל באמצעות טרד שקורא את תשובות מהשרת, ואת חלקן נציג בממשק הגרפי. הורדת קובץ מהשרת מתבצעת בצורה אמינה מעל UDP ומטופלת בנפרד, כלומר שהלקוח מבקש להוריד קובץ מהשרת נפתח טרד נוסף שמיועד רק לזה, ולכן הלקוח יוכל לבצע פעולות אחרות במקביל.

איתי קרואני 311504914
טל מלחי 208278556

הסבר על תהליך הורדת קובץ:
הורדת הקבצים מתבצעת בצורה אמינה מעל UDP . מימוש תהליך ההורדה נעשה בעזרת מנגנון GO BACK N , שיספק לנו אמינות בהורדת הקובץ בין השרת ללקוח. באפשרות כל לקוח לעצור את ההורדה ולהמשיך מאותה הנקודה. באפשרות כמה מחוברים לצ'ט להוריד קובץ במקביל, ולבצע פעולות נוספות אחרות בצ'ט במהלך ההורדה. במידה ולקוח ינסה להוריד שני קבצים במקביל, תופיע שגיאה כיוון שלכל לקוח סוקט אחד בלבד שמטפל בהורדת קבצים.

<u>מהצד של השרת:</u> הגדרנו חלון התחלתי בגודל 8 , בו השרת שולח ללקוח עד 8 חבילות. הגדרנו Timeout, בתור הזמן המקסימלי עבורו הלקוח יחזיר ACK עבור החבילה שהתקבלה בצד שלו. המנגנון עובד בצורה שהACK הוא מצטבר. כלומר, עבור ערך ACK מסוים נסיק כי כל החבילות עד אותו ערך הגיעו. במידה ולא הגיע ACK מהלקוח עבור חבילה מסוימת בחלון הנתון, נשדר מחדש עד שנקבל את הACK בחזרה.
ובנוסף, גודל החלון גדל בקצב אקספוננציאלי (פי 2 כל פעם). לאחר ההכפלה נבחר את גודל החלון להיות המינימלי בין גודל החלון הנוכחי לבין כמות הפאקטות נשארו לשלוח. בדרך זו אנו מתמודדים עם איבוד פאקטות ומוודאים שכל הפאקטות אכן הועברו בצורה אמינה.

<u>מהצד של הלקוח:</u> נגדיר משתנה expected , באמצעותו נוכל לדעת את מספר הפאקטה שאמורה להתקבל. עבור כל הודעה נכונה נשלח לשרת הודעת ACK ונשמור את כל הפאקטות במערך באמצעותו לבסוף נכתוב את הקובץ. במידה ולא קיבלנו את החבילה שצפינו לקבל, כלומר קיבלנו חבילה לא בסדר, נשלח לשרת הודעת Acknow-ledgment עם מספרה של הפאקטה אחת לפני אחרונה שהתקבלה.
בכל הודעת ACK נסתכל על רצף של סדרה שהתקבלה, וננסה לקבל שוב את החבילה שלא קיבלנו

אופן ההרצה: קיימות שתי אופציות:
1. במחשב אחד - להריץ את הפקודה python3 Server.py אשר תפעיל את השרת, ולאחר מכן את הפקודה python3 Client.py שתריץ את הלקוח- משני טרמינלים שונים
2. בשני מחשבים שונים- להריץ את הפקודה python3 Server.py אשר תפעיל את השרת ממחשב אחד, וממחשב שני, את הפקודה python3 Client.py IP PORT , כאשר IP וPORT מסמנים את פרטי השרת אותם נצטרך להכניס.
בהרצת הלקוח ייפתח הממשק הגרפי עם הכפתור start שבלחיצתו נתחבר לשרת ונוכל להשתמש בצ'ט.

קישור לגיט הפרויקט:https://github.com/itay-kar/Chat-and-files-server-TCP-UDP-connections.git
* לגיט מצורפים סרטון הרצה לדוגמא של הפרויקט וכן קבצי תעבורה מה wireshark