



עבודת גמר 5 יח"ל

נושא העבודה : פלטפורמת BitTorrent מורחבת

שם תלמיד : אלון לוי

ת.ז תלמיד : 214814683

שם בית ספר ועיר : קריית החינוך ע"ש עמוס דה-שליט, רחובות

שם המנחה : ערן בינט

מועד הגשה : 17/05/2023

תוכן עניינים

3	1. מבוא
5	2. תיאוריה
10	3. תוצר סופי
24	4. תהליך כתיבת הפרויקט
25	5. מרכיבי פתרון
33	6. תסריטי בדיקה
34	7. רפלקציה
35	8. הוראות התקנה ותפעול
35	9. ביבליוגרפיה
36	10. נספחים

1. מבוא

1.1. נושא העבודה

הפרויקט הינו תשתית רשתית Peer-to-Peer (P2P) להעלאת והורדת קבצי תוכן, המושתתת על הפרוטוקול BitTorrent. התשתית תאפשר הורדה מקבילית של קובץ תוכן או מספר קבצי תוכן על פי החלקים המרכיבים אותם, על בסיס המידע שיושג מה-torrent שיטען למערכת, ותוך טיפול מלא בשגיאות אפשריות. כחלק מהפרויקט מיושם ממשק משתמש גרפי (GUI) ידידותי המאפשר העלאה של קבצים לתשתית, ובנוסף מציג את ההתקדמות של כל אחד מה-torrents המורדים בכל עת.

המערכת מיישמת ממשק גרפי למטרת ניהול הרשת - Admin. כחלק מממשק זה, מנהל המערכת יכול לראות את כמות הפניות לשרת הניהול בכל עת, לצפות במצב המשתמשים המשתפים בקבוצות שונות ברשת הפנימית, לראות את המשתמשים החסומים וגם את ה-log המציג את הפעולות העיקריות שבוצעו במערכת, כדוגמת חסימת משתמשים.

במסגרת התשתית המוצעת, תתאפשר הורדה של חלקי קובץ המטרה מהרשת החיצונית, תוך יישום של הפרוטוקול BitTorrent, או מהרשת הפנימית תוך יישום מורחב של הפרוטוקול. במידה וחסר piece (חלק) אחד או יותר של קובץ המטרה בזמן ההורדה ברשת הפנימית, תתבצע פנייה לרשת החיצונית למציאת החלקים הרצויים תוך הסתייעות בתכנים הקיימים ב-BitTorrent, ובכך הסבירות להשגת קובץ המטרה תגדל.

1.2. מטרת מרכזיות

המטרות היישומיות של הפרויקט הן:

- יצירת תשתית עמיתים פרטית לשיתוף מידע
- מתן אפשרות להורדה מקבילית של חלקי קובץ ממספר עמיתים על מנת לנצל את מלוא רוחב הפס, ועל ידי כך - להשיג מהירות רבה יותר וניצול מקסימלי של התקשורת המקבילית.
- מתן כלי ניתוח ניהולי, ובכלל אלו סטטיסטיקות איכותיות ומדויקות לשרת הניהול (Tracker), העוקבות אחר המתרחש ברשת, כדוגמת מספר ההודעות שהשרת מקבל ממשתמשים בכל זמן קצוב, וכן סטטיסטיקות העוקבות אחר המשתמשים הנמצאים בכל קבוצה (Swarm).
- התאמת פתרון מיטבי לדרישות דינמיות של המשתמש, כולל טיפול בשגיאות תקשורת, וניצול מלא של תקשורת מקבילית במערכת, להורדת חלקי קבצים ולהעלאתם ללא השהייה.

המטרות האישיות של העבודה הן:

- למידת פרוטוקול BitTorrent לכל רוחבו על מנת שאוכל ליישם את רוב הדברים הכתובים בו וגם כדי להוסיף תכונות חדשות משלי באופן שלא יתנגשו עם המפרט הקיים.
- הרחבת הידע הנרכש בכיתה י"א בתחום תקשורת.
- מיצוי מלא של היכולות שלי כתלמיד במגמת סייבר, כך שהתשתית תעבוד מול תרחישים שאינם אידיאליים ובלתי צפויים.
- עמידה בלוח הזמנים כפי שנקבע עם המנחה.
- הרחבת הידע בכלים, בספריות ובשפות התכנות שנלמדו ולמידתם של אלו שלא נלמדו, באמצעות למידה עצמית.

1.3. [רציונל](#)

המוטיבציה שלי לפיתוח הרעיון הינה ההבנה שיש צורך בשדרוג הפרוטוקול BitTorrent על ידי הוספה של תכונות חדשות שיקלו על המשתמש להשיג את הקובץ אותו הוא דורש באופן פשוט ונוח. הבנתי גם שיש צורך גדול במיוחד במערכת שתאגד את כל הנמענים וההגדרות בפרוטוקול BitTorrent תחת תשתית אחת, ובנוסף תגדיר משתמש חדש מסוג מנהל (admin). תשתית כזו לא קיימת כיום, ופתרון זה ימצה את המיטב של כל אחד מהיבטים המוגדרים בפרוטוקול.

מטרה נוספת שלי היא שיפורו של שרת הניהול, המוגדר בפרוטוקול BitTorrent כך שיוכל לטפל בעצמו בשגיאות וגם יעביר תכונות מעקב ודיווח (שהוגדרו כשלו) למשתמש מסוג admin, על מנת להוריד מן העומס על שרת זה ולאפשר פעילות שרת רציפה.

בנוסף, רציתי לממש את הנלמד בכיתה י"א בנושא תקשורת הנתונים, ולהרחיב את הידע שלי ב-python, C++, ו-SQL.

1.4. [קישור לחומר הנלמד](#)

העבודה מתקשרת לחומר הנלמד במספר תחומים שונים.

- ראשית, הנושא אותו בחרתי הוא ה-highlight של החומר הנלמד בכיתה י"א. הפרויקט נועד לממש את כל הרעיונות אותן למדנו, כאשר נעשית עבודה משמעותית מעל sockets בפרוטוקולי UDP, TCP וגם HTTP. בנוסף, ימומשו גם ספריות שנלמדו בכיתה י"א כחלק מהרחבת הפרוטוקול, ובניהן matplotlib לתצוגה גרפית, SQLite למסד הנתונים ועוד.
- שנית, המערכת תוכל לאתר התקפות שונות, אשר נלמדו במסגרת כיתה יא' ולטפל בהן, לדוגמה: משתמש מציף רכיבים במערכת בהודעות שלא כצורך, או מנסה להעלות קובץ שגוי. בפרויקט יעשה גם שימוש רב ב-threads כחלק מהחומר הנלמד בתחום מערכת ההפעלה, כדי לאפשר מקביליות.
- תחום נוסף בפרויקט הוא תחום הגרפיקה. בפרויקט איישם את הגרפיקה באמצעות הספרייה PyQt, ספרייה אותה למדנו בכיתה י"א, ביישום הגרפיקה תידרש הבנה מעמיקה של הספרייה

מעבר לנלמד, כך שאמצה את המוצע בה באופן שיכסה את כל התחומים שהוגדרו בה, וביניהם סטטיסטיקות שרת הניהול אשר יוצגו בצורה גרפית ב-admin, תצוגת מצב הורדת הקובץ, וכו'.

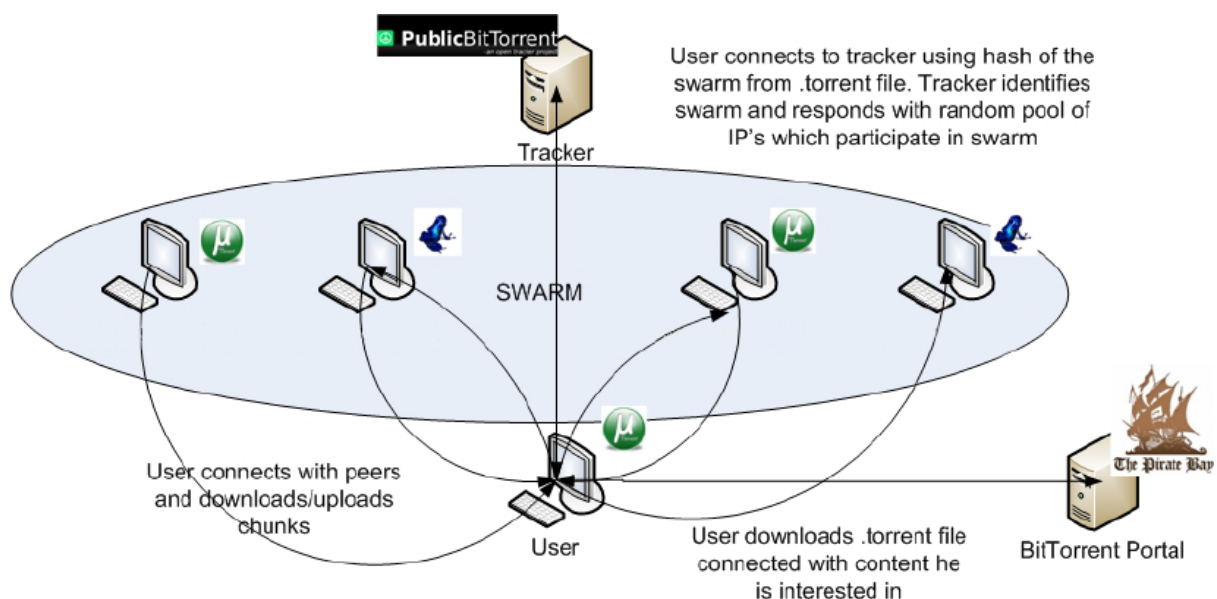
2. תיאוריה

2.1. תיאוריה

הפרויקט מתמקד בתשתית שיתוף קבצים מבוססת פרוטוקול BitTorrent.

BitTorrent הוא פרוטוקול תקשורת שנוצר בשנת 2001 על ידי בראם כהן, פרוטוקול זה פועל באופן של תקשורת בין עמיתים (P2P) לצורך הורדת קבצי תוכן מהאינטרנט. זהו הוא אחד הפרוטוקולים הנפוצים ביותר להעברת קבצים גדולים ברשת - בשנת 2013 היה אחראי ל-3.35% מתעבורת הרשת העולמית, ונכון להיום קיימים בו כ-250 מיליון משתמשים פעילים. הפרוטוקול נוצר מתוך ההבנה שלא קיימים שירותים אחרים שעובדים באופן P2P בצורה טובה, והוא נועד לשנות מצב זה.

כחלק מהפרוטוקול BitTorrent, מוגדר תהליך בסיסי, במסגרתו עמית פונה לשרת מידע חיצוני, לצורך השגת קובץ torrent. ממנו. על בסיס קובץ זה הוא מפענח מי הם שרתי הניהול אליהם יוכל לפנות ומקבל מידע על ה-pieces המרכיבים את קובץ התוכן הסופי. לאחר פענוח קובץ torrent, הוא פונה לשרתי הניהול ומהם משיג swarms שהם קבוצות של עמיתים המשתפים חלקים של קובץ התוכן הסופי, הוא פונה לעמיתים אלה ומוריד מהם pieces שמרכיבים את קובץ התוכן הסופי, הוא עושה זאת תוך כדי וידוא ה-pieces באמצעות המידע אותו פענח מקובץ ה-torrent. אותו הוריד, לבסוף מרכיב את pieces אלה לכדי קובץ תוכן אחד. **טיפטר מה leecher או תסביר נעשה**



תהליך זה נשען על מספר מושגי יסוד:

1. משתמש מערכת (User) – המשתמש הוא חלק מהמערכת הסגורה. תפקידו של המשתמש הוא להשיג קובץ מטרה, תוך איתורו בממשק המשתמש. המערכת בתורה תטפל בבקשה זו החל משליחת הבקשה ועד ההורדה המלאה של קובץ מטרה זה. משתמש גם יכול להעלות קבצי תוכן אותם ירצה לשתף עם משתמשים אחרים ברשת הפנימית דרך ממשק המשתמש.

2. קבצי תוכן – קובץ המטרה אותו המשתמש רוצה להשיג. כל קובץ תוכן בנוי מ-**pieces** שהם פיסות מידע של קובץ התוכן שנמצאות ברחבי הרשת. **pieces** הם בעלי גודל קבוע (פרט לאחרון) והם בנויים מ-**blocks**, גודל כל **block** מוגדר על ידי המערכת (בדור"כ גודל **buffer**), על מנת להשיג את הקובץ המלא, המשתמש יבקש מהעמיתים השונים ברשת **blocks** עד שיהיו ברשותו מספיק **blocks** ליצירת **piece**, כך יעשה עבור כל **piece** שבאמצעותם יבנה את קובץ התוכן השלם.

3. קובץ torrent - קובץ **Metadata**, המספק את המידע הדרוש להורדת קובץ המטרה השלם. את הקובץ משיג המשתמש משרתי מידע הנמצאים באינטרנט. הקובץ כולל:

- את כתובות שרתי הניהול החיצוניים (**Trackers** - יש מספר שרתי ניהול חיצוניים בקובץ **torrent**). ואת סוגם (**UDP/HTTP/WebTorrent**) בנוסף ל-**PORTS** שלהם.
- גודל קובץ / קבצי התוכן, ושם
- שם תיקיית הקובץ / קבצי התוכן
- גודל **piece**, למעט ה-**piece** האחרון (שגודלו איננו קבוע).
- רשימת ה-**hashes** של ה-**pieces** לצורך וידוא מהימנותם לאחר הורדה

ראה מבנה הקובץ **torrent**. בפרק ה **appendix (10)**.

4. עמית (Peer) – תפקידו של עמית הוא לשתף חלקי קובץ עם עמיתים אחרים, ובכך יוכל להרכיב קובץ תוכן שלם. עמית משתף חלקים וגם משיג אותם מהעמיתים האחרים ברשת. כל העמיתים יודעים על מצבם של כל העמיתים האחרים בכל עת, באמצעות מידע זה יודע העמית אילו חלקים הוא יכול להוריד מהעמיתים האחרים ברשת.

5. שרת ניהול (Tracker) – שרת אשר מנהל קבוצות עמיתים (**Swarms**) שונות בהתאם לקבצי ה-**torrent**. שנמצאים ברשותו (אותם השיג מהרשת או שהועלו על ידו ממשתמשים אחרים). שרת הניהול יכול לקבל קבצי **torrent**. ממשתמשים ובזאת ליצור קבוצות עמיתים חדשות. השרת יקבל את קבצי ה-**torrent**. מהמשתמשים ויעדכן את מסד הנתונים שלו תוך יצירה של **table** חדש ששמו יהיה שם קובץ ה-**torrent**, במידה והקובץ מתבסס על קובץ **torrent**. גלובלי ומשותף

ברשת הפנימית, סיומו יהיה "_LOC.torrent", ובמידה והועלה על ידי משתמש, סיומו יהיה "_UPLOAD.torrent".

6. Swarm – קבוצה אשר נוצרת על ידי שרת הניהול - התנאים להיווצרותה של קבוצה זו הם העלאה של קובץ בידי משתמש המערכת לשרת הניהול או יידוע של שרת הניהול על כך שמשתמש מערכת סיים להוריד קובץ גלובלי שנמסר ממנו (בסיום הורדה של קובץ גלובלי משרת הניהול המשתמש יעדכן את שרת הניהול). על כל קובץ torrent. שהועלה לשרת הניהול או קובץ גלובלי שהורד במלואו על ידי משתמש מערכת נוצר swarm, בכל swarm מוגדרים העמיתים שמשתפים את חלקי קובץ המטרה של ה-torrent. עליו מתבסס ה-swarm - רק כך ניתן לדעת בוודאות אילו עמיתים משתפים אילו קבצים.

כאשר שרת הניהול רוצה להוסיף משתמש חדש ל-swarm, הוא יבדוק האם קיים table בשם ה-torrent. שקיבל מהמשתמש. כאשר משתמש מתחבר לשרת הניהול ומבצע את תהליך בירור העמיתים (המחוברים ל-swarm) לגבי קובץ התוכן אותו הוא מעוניין להוריד, שרת הניהול יוסיף את זיהוי המשתמש ל-swarm, וייצור רשומה חדשה שלו ב-table המתאים במסד הנתונים. שרת הניהול יודע מי מחובר לקבוצה, והוא מנטר את המתרחש בכל ה-swarms עליהם אמון.

7. Database – מסד נתונים משותף בין שרת הניהול ל-admin, הנמצא במערכת בה נמצא שרת הניהול. במסד נתונים זה מתבצע עדכון של כל המתרחש ב-swarms עליהם אחראי שרת הניהול, וכן נשמר המידע על כל המשתמשים הנמצאים ב-swarm בכל רגע, כך שלדוגמה במידה ומשתמש יפנה לשרת הניהול על מנת להצטרף ל-swarm מסוים, ישיג שרת הניהול את כל הרשומות של המשתמשים הנמצאים ב-swarm הנדרש, ישלח לו את ה-IP וה-PORT של משתמשים אלה, ויוסיף רשומה של המשתמש למסד הנתונים כעמית נוסף הנמצא ב-swarm. כחלק מערכי רשומה במסד נתונים זה, נכללים ה-IP וה-PORT של העמית, והזמן בו נוספה הרשומה (לצורך ניטור והסרה במידה ולא שיתף). שרת ניהול יכול לנהל מספר swarms, כאשר כל swarm הוא table במסד הנתונים של שרת הניהול.

8. מנהל (admin) - הגדרה חדשה כחלק מהפרויקט שכן פרוטוקול BitTorrent לא מגדיר אותו. ל admin יש יכולות ניהול המאפשרות לו לעקוב אחר המתרחש ב-swarms אותם מנהל שרת הניהול ולחסום עמיתים מ-swarms השונים בהתאם. לרשות ה-admin גם תצוגה גרפית של כמות הפניות לשרת הניהול בכל זמן קצוב. באמצעות שימוש בכלי זה, ה-admin יזהה את המשתמשים אשר מבצעים שליחה מופרזת של הודעות לשרת הניהול, ויחסום אותם מהתקשרות נוספת עם שרת הניהול.

2.2. מוצרים קיימים

כיום בשוק לא קיימות מערכות המיישמות את כל המוגדר בפרוטוקול BitTorrent (משתמש, שרת ניהול (Tracker)) בתשתית אחת, אלא רק במסגרת מערכות מפוצלות, וביניהן:

qBitTorrent הוא ממשק לקוח של הפרוטוקול BitTorrent שמאפשר הורדה של קבצים. הממשק מקבל קובץ torrent, שולף ממנו את המידע הנדרש (trackers, files, pieces) ומתחיל בהורדה. התוכנה מאפשרת להוריד את הקבצים בצורה מקבילית, מטפלת בשגיאות תוך הורדת הקבצים באמצעות האלגוריתם "rarest piece" שמוריד מה-piece הנדיר, זה שבבעלות הכי פחות מהעמיתים בקבוצה לנפוץ, שבבעלות מספר העמיתים הרב ביותר. התוכנה מאפשרת למשתמש להגביל את רוחב הפס שיהיה בשימוש על ידה, וגם תומכת ב-plugins לדוגמת מנוע חיפוש של שרתים המחזיקים קבצי torrent, וזהו הערך המוסף שלה כאשר BitTorrent כפרוטוקול כשלעצמו אינו תומך במנוע חיפוש. ממשק זה מיישם בצורה מלאה את הפרוטוקול מנקודות המבט של המשתמש והעמית בלבד, ללא יישום של שרת ניהול. כמובן, בממשק אין יישום של admin שכן זה ממשק של משתמש BitTorrent בודד.

thepiratebay הוא שרת הניהול (Tracker) המוכר והשמיש ביותר היום, עם 5,100,000 (!) עמיתים מחוברים בו זמנית, וניהול swarms של כ-630 אלף torrents שונים, שרת ניהול זה לא עוקב אחר המתרחש ב-swarms אותם הוא מנהל, מעבר למעקב של אילו עמיתים משתפים ואילו לא. השירות אינו קוד פתוח אך כולל סטטיסטיקות כחלק מממשק ניהול כרוב שרתי הניהול. ישנם שרתי ניהול (Trackers) רבים ומגוונים בשוק, כאשר רובם מיושמים בשכבות 4 ו-7 במודל ה-OSI, שרת ניהול זה מיושם בשכבה 7, תוך שימוש בפרוטוקול HTTP.

מערכות שיתוף קבצים נוספות קיימות, ובניהן המערכת **eDonkey**. לקוחות של מערכת זו נמצאים היום בשימוש קטן בהרבה מאשר אלו של BitTorrent, זאת משום שמהירות המערכת בהשוואה למערכת BitTorrent היא איטית, ולמרות זאת, במערכת שיתוף קבצים זו קיים מנוע חיפוש מובנה, אשר לא קיים ב-BitTorrent, ייחוד זה מושך משתמשים להשתמש בממשקי משתמש של מערכת שיתוף קבצים זו על פני שימוש בממשקי משתמש מושתתי BitTorrent.

המערכת שלי ייחודית במספר מובנים:

במרבית המוצרים הקיימים בשוק, כדוגמת BitTorrent, חסר מרכיב ה-admin ולכן כל דרישות הניהול נופלות על שרת הניהול (Tracker), כולל האחריות לחסום משתמשים, לעקוב אחר מצב ההורדה וההעלאה של pieces אצל כל אחד מן המשתמשים, לאור נסיבות אלה, אישם בפרוטוקול שלי גם admin, כחלק מהמערכת הסגורה, ברשת הפנימית, ואפשר:

- לחסום משתמשים ספציפיים, בצורה ידנית.

- לנטר את כמות הבקשות שנשלחות לשרת הניהול בכל 5 שניות ולחסום משתמשים אשר ישלחו 10 בקשות או יותר בטווח זמן זה.

אומנם קיימות מערכות שיתוף קבצים אחרות שכן מיישמות את התשתית, eDonkey היא אחת מהן, או מערכת שמרחיבה את הפרוטוקול BitTorrent לדוגמת הוספת admin לפרוטוקול, אך מעטות מאוד המערכות הכוללות ממשק משתמש ושרת ניהול, וגם הן חסרות את כל הנלווה להם, כמפורט בפרויקט זה.

ניתן לייצג את ההבדלים בטבלה:

Edonkey	thepiratebay	qBitTorrent	הפרויקט שלי	
V		V	V	הורדה והעלאה מקבילית של קבצים לפי pieces
V	V	V	V	GUI ידידותי למשתמש ו/או לשרת הניהול
V	V		V	יישום שרת ניהול (Tracker) כולל סטטיסטיקות
V		V	V	טיפול בשגיאות תקשורת בצד המשתמש
V		V	UDP, HTTP ONLY	תמיכה בסוגי שרתי הניהול השונים (, UDP, HTTP WebTorrent trackers)
			V	יישום admin ברשת הפנימית
	V	V		הביצועים הטובים בשוק
			V	יכולת הרחבה של המערכת על ידי ממשק חיצוני
V			V	מנוע חיפוש קבצי torrent. מובנה במערכת

3. תוצר סופי

3.1. תיאור הפרויקט

המערכת היא יישום מלא של הפרוטוקול BitTorrent עם כל הנלווה לו. היא מאפשרת הורדה של קבצי תוכן לפי הפרוטוקול BitTorrent, ועובדת גם ב-LAN, וגם ברשת החיצונית. המערכת כאמור מציעה הרחבה של הפרוטוקול באמצעות הוספת פונקציית admin וכן הודעות נוספות שלא הוגדרו בפרוטוקול כמו עדכון מצד העמית לשרת הניהול כשסיים להוריד את קובץ התוכן אותו ביקש. המערכת מספקת למשתמש כלים מתקדמים ומציגה בפניו מידע מפורט על ההורדה על מנת לאפשר נוחות משתמש מקסימלית, ובצד ה-admin מציגה בפניו גרפים על המתרחש בשרת הניהול.

רכיבי המערכת העיקריים:

- עמית (Peer) – יוזם לעמיתים אחרים בהתאם ליכולתו להשיג קובץ וגם לקבל עזרה מאחרים להורדת קובץ המטרה ומעדכן את מצב הורדת החלקים שלו עם העמיתים האחרים בקבוצה.
- Tracker (שרת הניהול) – מנהל קבוצות תקשורת המודע לכל המתרחש בכל הקבוצה, המשתמשים המוגדרים בה, ואילו pieces הם חולקים, ולרשותו גם סטטיסטיקות של כמות הפניות שהתקבלו על ידו בכל 5 שניות, אותן משתף עם ה-Admin. נתונים אלו מהווים הרחבה לפרוטוקול הקיים ב-BitTorrent, שכן שרת ניהול כמוגדר בפרוטוקול המקורי אינו מודע למתרחש בתוך הקבוצה, אלא רק לגבי המשתמשים הפועלים בה. כל התהליכים בשרת זה הם **אוטומטיים**. לדוגמה, אם עמית ישלח הודעות מרובות לשרת הניהול בטווח זמן קצר (10 הודעות ב-5 שניות), השרת יחסום את המשתמש אוטומטית מהתקשרות נוספת איתו. תפקידו העיקרי של ה-Tracker הוא להוסיף עמיתים חדשים לקבוצות שונות, תפקיד נוסף של שרת זה הוא קבלת קבצי torrent. ממשתמשים לצורך יצירת swarms חדשים דרך שימוש במסד הנתונים המשותף בינו לבין ה-admin.
- admin – ה-admin יודע על המתרחש בכל הקבוצות בשרת הניהול באמצעות המידע אותו מספק לו שרת הניהול דרך מסד הנתונים המשותף, לרשות ה-Admin יכולות חסימת משתמשים ידנית, ותפקידו לוודא שהכול בקבוצה הולך כשורה. לרשות ה-Admin גם סטטיסטיקות גרפיות של כמות הפניות אצל שרת הניהול, ו-log המאפשר לראות מה עשה ה-admin.

מרבית התקשורת המבוצעת בפרויקט הינה P2P (לדוגמה בין עמיתים ברשת), עם זאת, המערכת מיישמת גם תקשורת Client-Server,

- בין המשתמש שמעוניין להשיג את קובץ המידע (torrent). לבין שרת המידע החיצוני המספק לו את אותו הקובץ (HTTP Client-Server),

- בין עמית שמעוניין להצטרף ל-swarm לבין שרת הניהול שמספק לו את המידע לגבי המשתמשים בקבוצה אליה מעוניין להצטרף (UDP Client-Server)
- בין משתמש שרוצה להעלות קובץ מידע לשרת הניהול (TCP Client-Server)
- בין עמיתים ברשת הפנימית המשתפים חלקי קבצי תוכן בתקשורת P2P מקבילית

לאחר שמשתמש מפעיל את ה-GUI, הוא מציין בשורת החיפוש מה ברצונו להוריד, והמערכת מאתרת קובץ torrent. תואם מהרשת (החיצונית) וטוענת אותו. המערכת בוחנת את קבצי ה-torrent. שהושגו ומוציאה מהם את רשימת ה-trackers, פונה אל כולם באמצעות אלגוריתם להשגת כל העמיתים מכל הקבוצות, ומתחילה בהורדה מהעמיתים השונים בהתאם ליכולתם, תוך השוואת כל piece מהחלקים שמרכיבים את הקובץ ל-hash המתאים בקובץ torrent. שנטען.

שרת הניהול (Tracker) שומר את המידע לגבי כל משתמש והשעה בה נוסף במסד הנתונים.

המערכת מאפשרת מספר פעולות:

- הורדת קובץ ממספר עמיתים כאשר כל עמית אחראי על pieces הנמצאים ברשותו והמרכיבים קובץ מטר, ולבסוף חיבור ה-pieces לקובץ השלם בגמר ההורדה.
- הורדה מה-piece הנדיר ביותר לאלו שפחות כדי לאפשר הורדה רציפה.
- הורדת מספר קבצים בצורה מקבילית.
- הצגת מצב ההורדה בצורה אינטואיטיבית. המשתמש יראה לאורך כל תהליך ההורדה את ה-peers הזמינים ומאילו הוא מוריד חלקים, הודעות הכוללות עמיתים איתם הוא נתקל בשגיאות במהלך ההורדה ואת מצב ההורדה.
- העלאת קבצי torrent ל-tracker וחסמת משתמשים במידה ונשלח קובץ שאיננו תקין.

אם לא ימצאו pieces במערך הצמתים הפנימי, תתבצע פנייה למשתמשים ברשת החיצונית שברשותם חלקים נדרשים של הקובץ ובכך יתאפשר להשיג את אותם ה-pieces, כמובן, תוך שימוש בפרוטוקול BitTorrent.

המערכת מאפשרת הפקת סטטיסטיקות בשרת ניהול (tracker) ותצוגה איכותית של סטטיסטיקות אלה בצד ה-admin, כמו גם אפשרות לראות את המשתמשים החסומים, ולחסום באופן ידני משתמשים מקבוצות לפי הצורך.

3.2. אלגוריתמים עיקריים

אלגוריתמים מרכזיים במערכת:

1. מציאת סטטוס של קובץ

זהו השלב הראשון בתהליך השגת קובץ תוכן בידי משתמש. אלגוריתם זה אחראי על פענוח קובץ ה-torrent, את קובץ זה המשתמש ישיג באמצעות:

- פנייה לשרת מידע חיצוני או פנימי עם שם הקובץ אותו רוצה להוריד
- טעינה ידנית של הקובץ

כל קובץ torrent. כתוב בקידוד "bencode" כ-dictionary של ערכים שונים כמוגדר בפרוטוקול BitTorrent, כך שעל מנת לקשר בין הנתונים השונים בקובץ זה, יש צורך בידיעת הקידוד. בעזרת ספרייה ב-python ששמה הוא bencode.py דבר זה נעשה פשוט הרבה יותר - ולמרות זאת, קובץ ה-torrent. חסר בתוכן, כך שיש צורך גם ביישום אלגוריתם.

להלן דוגמה לאופן בו האלגוריתם יכול לבוא לידי ביטוי:

בקובץ torrent. לא נתונים כל הגדלים של כל קבצי המטרה, בנוסף, ה-hashes של ה-pieces שכתובים בקובץ זה מסודרים בסדר כרונולוגי, כך שבמידה וקיים יותר מקובץ מטרה אחד בקובץ זה, לא יהיה ניתן לדעת אילו קבצים מחזיקים אילו חלקים, כאן מגיע האלגוריתם לידי ביטוי, הוא ישייך חלקים לקבצים כך שברגע שיושג החלק הוא ייכתב בקובץ המתאים ב-offset המתאים באותו קובץ מטרה.

סדר פעולות האלגוריתם:

- פתיחת קובץ ה-torrent. כפי שנטען למערכת (השגתו מפורטת לעיל)
- פענוח הקובץ בעזרת הספרייה bencode.py, הוספה למבנה נתונים dictionary
- שמירת נתונים שונים שלא יכולים להימצא בקובץ במבני הנתונים המתאימים להם במחלקת המשתמש

2. מציאת עמיתים לצורך שיתוף חלקים, כולל לצורך השלמת חלקים מהרשת החיצונית

זהו השלב השני בתהליך השגת קובץ תוכן בידי משתמש. המערכת תפנה בצד המשתמש לשרת הניהול הפנימי ותשאל אותו האם קיימת קבוצה המשתפת את ה-torrent שהושג אצלו, במידה ושרת הניהול ימצא שכן, המערכת תברר עם כל העמיתים מה הם החלקים שברשותם, תשמור אותם במחלקת המשתמש ותתקדם לשלב ההתחברות אליהם אך במידה ולא, או יתברר בתהליך בירור שלא כל ה-pieces נמצאים ברשת הפנימית, תחל המערכת בפעולות הבאות, לצורך חיבור לעמיתים ברשת החיצונית:

- תשתמש בנתוני קובץ ה-torrent. שנטען לצורך השגתם של ה-trackers עליהם מורה
- תתחבר לכל tracker בהתאם לטווגו (HTTP, UDP) ותשלח בקשה לכל tracker בקובץ להשגת משתמשים משתפי הקובץ (משתמשי הקבוצה אצל ה-tracker)
- תקבל את רשימת העמיתים מכל ה-trackers, ותשמור אותם במחלקת המשתמש, תמשיך לשלב ההתחברות אליהם

3. מציאת העמית הרלוונטי להרכבת קובץ מטרה, וקישור בין עמיתים והחלקים שברשותם
 זהו השלב השלישי בתהליך השגת קובץ תוכן בידי משתמש. האלגוריתם אחראי על הקישור בין העמיתים לבין החלקים שברשותם, סדר פעולותיו:

- האלגוריתם ישלח הודעות "handshake" לכל העמיתים שנמצאו באופן מקבילי
- יקבל הודעת "bitfield" מכל עמית, בהודעה זו כתוב אילו pieces בבעלותו של העמית
- יעדכן את רשימת ה-pieces הזמינים ובידי אילו עמיתים בקבוצות של הקובץ שנטען

4. אופן התקשרות עם עמיתים כולל טיפול בשגיאות
 זהו חלק א' מהשלב הרביעי והאחרון בתהליך השגת קובץ תוכן בידי משתמש. אלגוריתם זה מגדיר את האופן בו תתבצע ההתחברות לעמיתים שונים בידי משתמש המערכת לצורך הורדת חלקים, לכמה משתמשים יתחבר במקביליות, לאילו ובאיזה אופן. לאחר הקישור בין העמיתים לחלקים שברשותם, יחל בסדר הפעולות הבאות:

- יעבור על כל ה-pieces השונים מהנדיר לנפוץ ביותר, בלולאה
- על כל איטרציה בלולאה זו:

- יבדוק האם התגלו שגיאות באחד או יותר מההקצאות באיטרציות הקודמות, במידה וכן יבצע את השלב הבא עבור כל piece עבורו התגלתה שגיאה, לפני שיעבור ל-piece עבורו לא הוקצתה תקשורת
- יפתח תקשורת עם עמית רנדומלי מרשימת העמיתים שמחזיקים את ה-piece ב-thread, ימשיך לעשות זאת עד שישייך piece לכל עמית זמין ברשת
- יחכה בלולאה פנימית למקום פנוי נוסף, כלומר עד אשר יסיים עמית להוריד את ה-piece ששוין אליו, ימשיך בפעולות אלה עד לסיום כל תהליכי התקשורת הפתוחים.

5. תקשורת בין עמיתים לצורך השגת חלקים
 זהו חלק ב' מהשלב הרביעי והאחרון בתהליך השגת קובץ תוכן בידי משתמש. באמצעות האלגוריתם מתבצעת התקשורת המורה על בקשת המשתמש מהעמית אליו התחבר,

כמוגדר בפרוטוקול BitTorrent, בנוסף להורדת piece מאותו עמית. ההתחברות והצהרת כוונות לעמית נראות כך:

- התחברות משתמש המערכת לעמית באמצעות לקוח TCP בעל port רנדומלי
- שליחת הודעת "Handshake" המורה על התחלת שיחה בין עמיתים
- קבלת הודעת "Handshake" מורחבת מהעמית המחובר, שמהווה אישור להתחלת שיחה
- שליחת הודעת "interested" המורה על רצון להוריד pieces, שולח הודעה זו פעם אחת בתחילת השיחה בשלב זה בלבד
- קבלת הודעת "unchoke" מהעמית המבטאת את כך שעמית זה זמין לשיתוף של pieces
- תחילת הורדה באמצעות שליחת הודעות "request" של כל block שמרכיב את ה-piece שהוקצה להורדה, לאחר ששיג משתמש המערכת את כל ה-blocks שמרכיבים את ה-piece, יבנה את ה-piece במידה וה-hash שלו תואם לזה שנמצא בקובץ torrent, ישמור אותו אצלו ויסיים תקשורת

6. בניית קבצים לפי החלקים שאוחזו

זהו חלק ג' מהשלב הרביעי והאחרון בתהליך השגת קובץ תוכן בידי משתמש. האלגוריתם מאפשר בנייה של קבצים לפי ה-pieces שהורדו. בעת סיום הורדת חלק, האלגוריתם יכתוב את המידע של אותו החלק ב-offset המתאים בקובץ המתאים, בעזרת המידע שהושג לגבי היכן לכתוב באלגוריתם מציאת הסטטוס של קובץ.

7. טיפול בתקלות

מדובר ביישום של מספר אלגוריתמים, תפקיד כל אלגוריתמים הוא להתמודד עם תקלות בתחומים השונים שבפרויקט.

- **בתחום התקשורת בין עמיתים:** במידה ובוצע ניסיון תקשורת בין המשתמש לעמית אחר בקבוצה, והעמית מסרב לספק את התוכן הדרוש (Timeout או הודעת choke), לדוגמה ניסיון הורדת חלק מעמית כאשר העמית שולח הודעת choke שנועדה לשדר חוסר רצון לשלוח pieces (בגלל כמות רבה של עמיתים שכבר מחוברים לעמית זה) לאחר הודעת interested, יבוא אלגוריתם זה לידי ביטוי, ויבצע את הדברים הבאים:
 - יסגור את thread התקשורת הנ"ל, ויבדוק האם ניתן לפתוח thread חדש שינהל את הורדת ה-piece הדרוש, כלומר האם המכסה לכמות ה-threads האחראים על הורדת חלקים במקביליות מלאה, במידה וכן יחכה למקום פנוי

- יבצע בקשת piece נוספת מעמית אחר שהצהיר בהודעת ה-"Handshake" שבבעלותו ה-piece הנ"ל

- במידה והעמית אליו מחובר משתמש המערכת איננו מגיב, תבוצע בקשה חוזרת של block (הודעת request) מהעמית, במידה וגם לאחר הודעה זו לא יתקבל התוכן הנדרש, יחל בסדר הפעולות המפורט לעיל.
- לאחר הורדת חלק מעמית ה-hash של ה-piece המורד איננו תואם ל-hash שבקובץ ה-torrent, תבוצע בקשה חוזרת של כל ה-piece מהעמית, ובמידה וגם לאחר בקשה זו תהיינה אי תאימות, יחל בסדר הפעולות המפורט לעיל.

8. זיהוי התנהגות לא נאותה ברשת

אלגוריתם זה יבוא לידי ביטוי בשרת הניהול וב-admin, תפקידו הוא למנוע ממשתמשים לנסות ולפגוע ביציבות המערכת.

האלגוריתם בשרת הניהול: כאשר משתמש "מציק" לשרת הניהול, כלומר שולח לו הודעות חוזרות ונשנות שלא כצורך, האלגוריתם יזהה בעזרת סטטיסטיקות שרת הניהול את הנעשה, ויחסום באופן אוטומטי את המשתמש מכל התקשרות נוספת עם שרת הניהול. אופן נוסף בו האלגוריתם יכול לבוא לידי ביטוי הוא בתחום העלאת קבצי ה-torrent. אליו בידי משתמשים. כחלק מהפרויקט תתאפשר גם העלאת קבצי torrent. לשרת הניהול לצורך יצירת Swarm חדש (של שיתוף הקובץ שהועלה). במידה והקובץ אשר הועלה לשרת הניהול פגום, ייחסם המשתמש באופן אוטומטי מכל התקשרות נוספת עם שרת הניהול הזה, זאת על מנת למנוע ממשתמשים להרוס את שרת הניהול בכך שישותפו קבצי torrent. שאינם ניתנים לפענוח.

ה-admin עוקב אחר הנעשה בקבוצות השונות עליהן אחראי שרת הניהול, משום שהוא בעל גישה למסד נתונים משותף בינו לבין שרת הניהול. באמצעות נתונים ממסד נתונים זה, יודע ה-admin על מצבם של כל העמיתים בקבוצה. ביכולתו לחסום משתמשים ידנית, כלומר למחוק אותם מה-swarm ולחסום אותם מהתקשרות נוספת עם שרת הניהול, או "לנתק" אותם, כלומר למחוק אותם מה-swarm בלבד.

3.3. דרישות ומגבלות מערכת

למערכת מספר דרישות ומגבלות.

בהיבט הדרישות:

- המערכת חייבת להיות מהירה, וחייב להיות ניצול מלא של רוחב הפס, הורדה מקבילית כל הזמן מכל העמיתים מהם אפשר להוריד

- מסד הנתונים המשותף בין ה-admin לשרת הניהול חייב להיות פעיל תמיד, זאת על מנת שיהיה ניתן לקבל תמונה רחבה ועדכנית של המתרחש בקבוצות, וגם לוודא ניצול אופטימלי של האלגוריתם לחסימת משתמשים מקבוצה על בסיס אופן העזרה שלהם למשתמשים אחרים אצל המנהל.
- המערכת חייבת לתמוך בהורדת חלקים מן הרשת החיצונית על מנת לאפשר הורדה מלאה של קובץ ובאופן רציף.
- תמיכה בכל סוגי קבצי התוכן
- טיפול משמעותי במשתמשים בידי ה-admin, כחלק מהרחבתו של הפרוטוקול BitTorrent.

בהיבט המגבלות:

- ראשית, הפרוטוקול BitTorrent המורחב לא יעבוד ברשת החיצונית, זאת מכיוון ש-trackers ו-peers ברשת החיצונית משתמשים בפרוטוקול BitTorrent ללא ההרחבות אותן אישם, אך בתשתית יהיה ניתן גם להוריד קבצים מהרשת החיצונית ולא רק ב-LAN על ידי מעבר פשוט לפרוטוקול BitTorrent.
- בנוסף, ההגבלה העיקרית היא שבפרויקט שלי לא תהיה תמיכה בכל סוגי ה-trackers, בשל הכמות העצומה של trackers בעלי סוגים שונים, מגבלה זו מונעת מהמערכת לפנות ל-trackers מסוימים שכן לכל סוג tracker נוסח פנייה אחר, אך אתמוך בעיקריים שביניהם – UDP, HTTP, וה-tracker אותו אישם כחלק מהרשת הפנימית יהיה UDP.
- רק admin אחד יוכל להתחבר בכל עת לכל שרת ניהול, בשל כמות האפשרויות הקיימות ב-admin, והאפשרות שיוצרו התנגשויות.
- תמיכה במערכות הפעלה מסוג Windows 7, Windows 10 בלבד
- רצוי ש-admin יהיה פעיל תמיד ביחד עם שרת הניהול, כדי לאפשר שה-local torrents יהיו מעודכנים בכל עת בנוגע לעמיתים הנמצאים בקבוצה, ובכך למנוע פניה מיותרת אליהם בידי משתמש המערכת

3.4 ממשקים למערכות חיצוניות

- המערכת תהיה מושתתת על הפרוטוקול BitTorrent, פרוטוקול P2P לשיתוף קבצים, תפקיד הפרוטוקול הוא לסייע בהעברת קבצים בין עמיתים ברשת, גרסת הפרוטוקול: 1.0.
- תיעשה התממשקות מלאה לפרוטוקול כאשר תהיה פנייה לרשת החיצונית, ברשת הפנימית ייעשה שימוש ברכיבים שהוגדרו על ידו, אך עם שינויים ותוספות.
- הגדרות הפרוטוקול בהם ייעשה שימוש:

- Trackers

- Peers
- קובץ torrent.
- Peer wire protocol – אופן התקשורת בין עמיתים (אילו הודעות לשלוח, למי, מתי)
- Bencoding
- Rarest First piece downloading strategy – הורדת pieces מה-piece הנדיר, שבבעלות הכי פחות עמיתים, ל-piece הנפוץ, שבבעלות הכי הרבה עמיתים.

ייעשה גם שימוש ב-API של x1337 לצורך חיפוש torrents כחלק מתכונת החיפוש בממשק המשתמש, במידה ולא נמצא קובץ ה-torrent. ברשת הפנימית (משרת המידע הפנימי).

3.5. [התייחסות לנושא אבטחה](#)

בפרויקט מספר התייחסויות לנושא אבטחה:

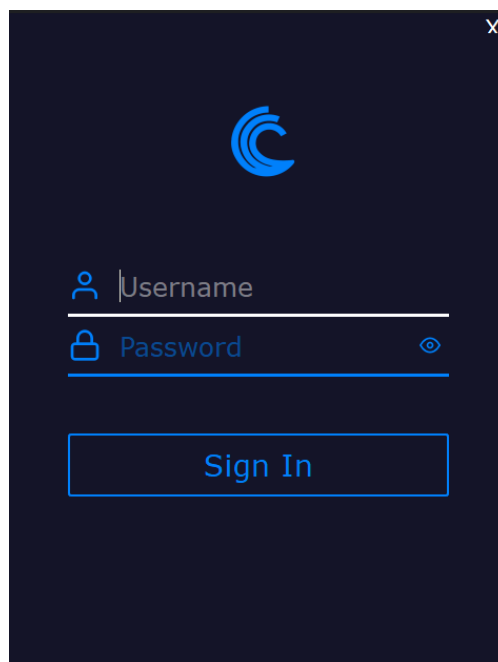
- כניסה למערכת כ-admin תדרוש שם משתמש וסיסמה
- תיעדוף עבודה ברשת הפנימית – פנייה לרשת החיצונית רק במידה ולא נמצא torrent בשרת הניהול הפנימי, או לא כל ה-pieces נמצאים ב-swarm הפנימי
- תקשורת SSL בין מרכיבי הרשת הפנימיים

3.6. [ממשק משתמש](#)

Admin

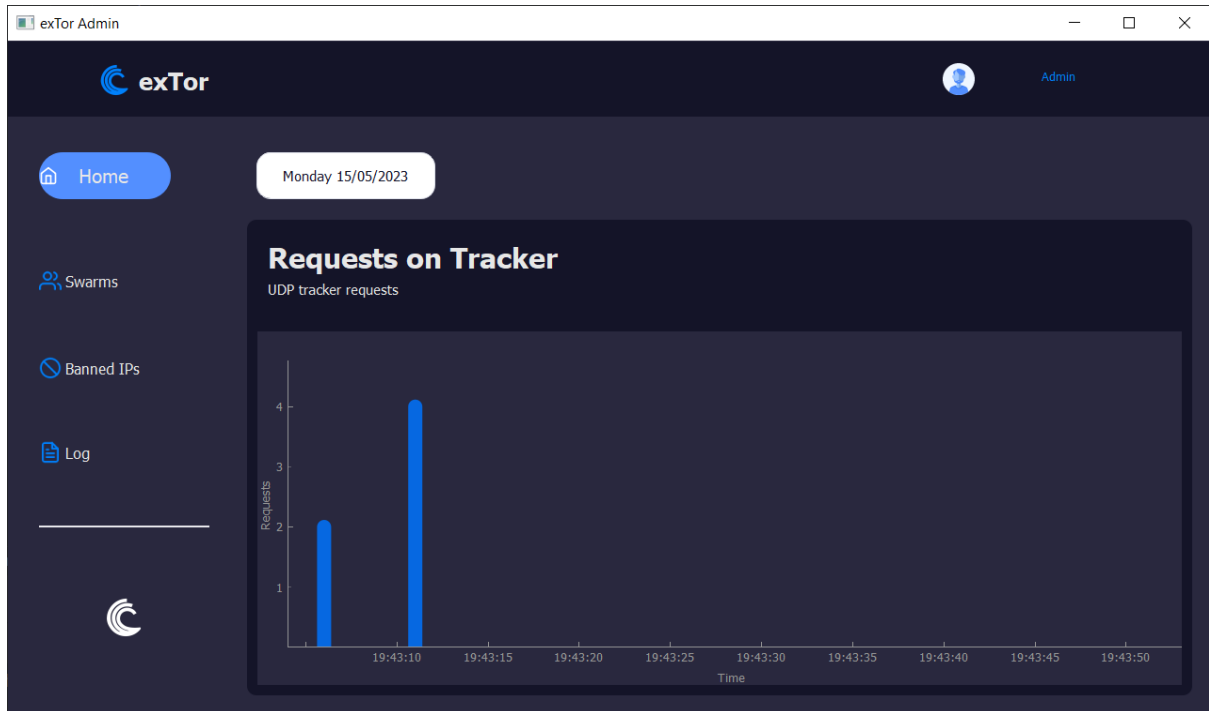
1. חלון כניסה:

בהרצת ה-Admin, יופיע חלון ההתחברות הבא:



The image shows a dark-themed login window for the Admin interface. At the top center is a blue circular logo with a stylized 'C'. Below the logo are two input fields: the first is labeled 'Username' with a person icon, and the second is labeled 'Password' with a lock icon and a toggle eye icon. At the bottom center is a large button labeled 'Sign In'.

2. חלון ראשי (חלון סטטיסטיקות פניות לשרת הניהול):
לאחר התחברות יופיע החלון הראשי של האפליקציה:



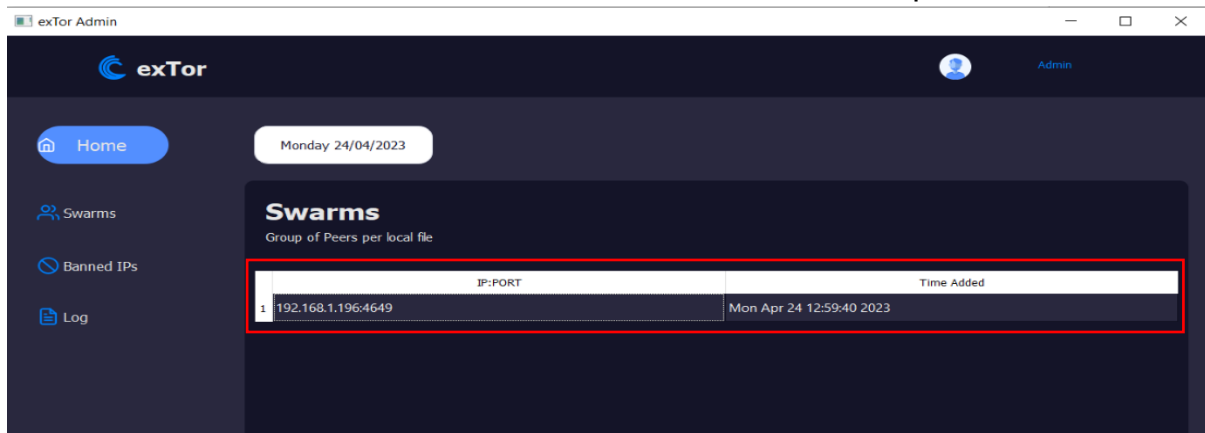
במרכז חלון זה ניתן להבחין בכמות הפניות לשרת הניהול בכל 5 שניות בתצוגה גרפית מתעדכנת, מאחורי הקלעים ה-Admin מפענח את הפניות ומי ביצע אותן ובמידה ויזהה פניות רבות לשרת הניהול (במקרה הזה 10), יחסום את השולח מהתקשרות נוספת עם שרת הניהול.

3. חלון Swarms:

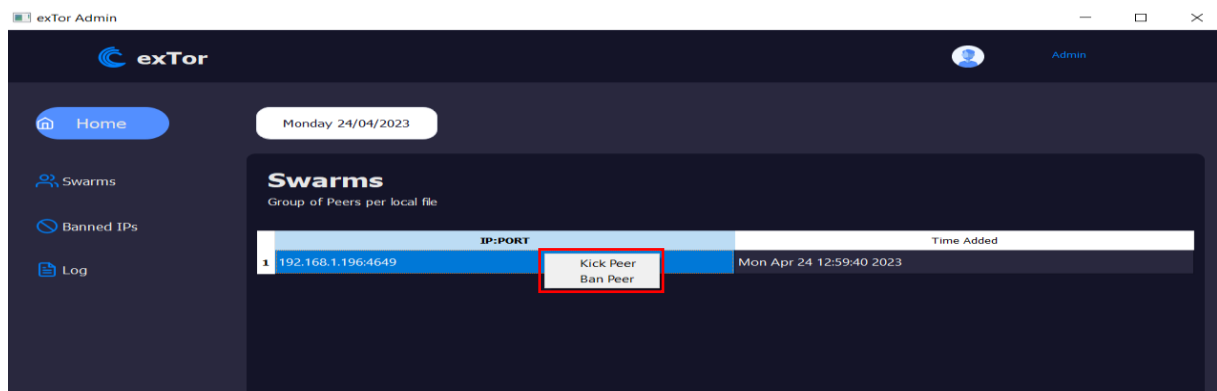
בתפריט השמאלי ניתן להבחין ב-Swarms, במידה ונלחץ על כפתור זה יוצג בפנינו מסד הנתונים המשותף של שרת הניהול וה-Admin:

	File Name
1	TestFile_UPLOAD.torrent

במקרה זה ניתן לראות כמסומן באדום שמשתמש העלה לשרת הניהול קובץ בשם TestFile, זאת משום שלצד שם הקובץ כתוב "_UPLOAD". קובץ זה מופיע כרשומה במסד הנתונים המשותף, לחיצה כפולה עליו תציג:



מסך זה יציג לנו (כמסומן באדום) את כל העמיתים הנמצאים ב-Swarm ומשתפים את הקובץ. במקרה זה ניתן לראות שמחובר העמית 192.168.1.196:4649, בעמודה השניה ניתן לראות את התאריך והזמן בו התחבר (24 באפריל 2023 בשעה 12:59), מקש ימיני על רשומה זו יציג:

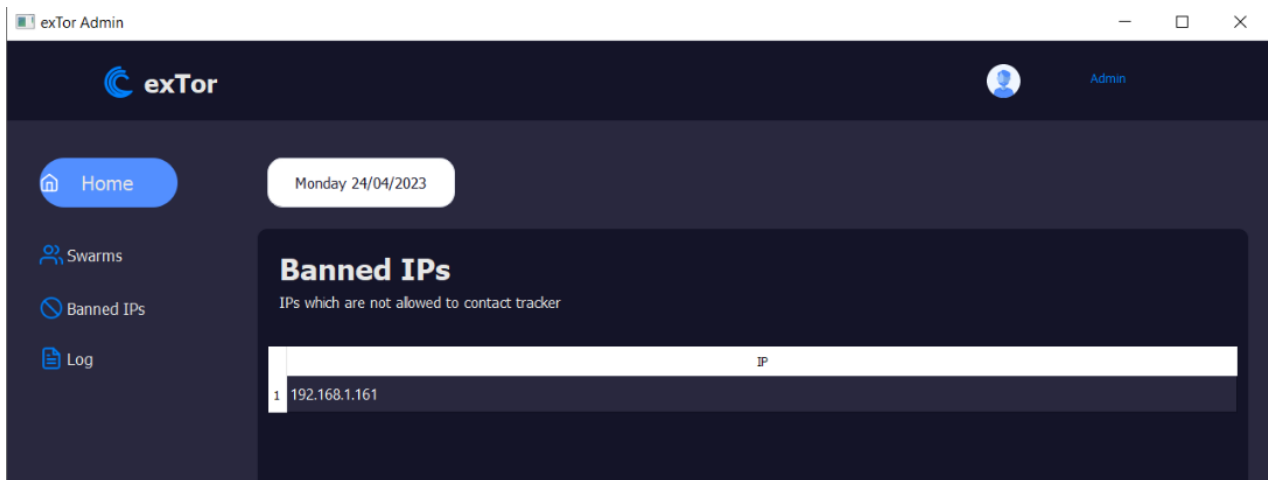


לפנינו יופיעו 2 אפשרויות (כמסומן באדום):

- Kick Peer - ניתוק המשתמש מה-Swarm בו הוא נמצא
- Ban Peer - חסימת המשתמש כך שינותק מה-Swarm ולא יוכל לבצע פעולות ברשת הפנימית יותר. (במקרה זה החלטתי "להעיף" את המשתמש בלבד.)

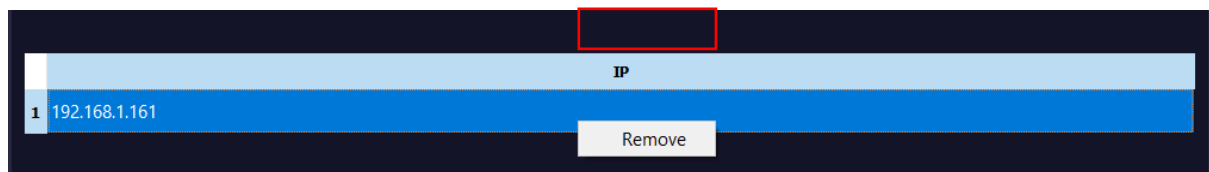
4. חלון Banned IPs:

השורה הבאה בתפריט השמאלי היא Banned IPs, במידה ונלחץ על כפתור זה תיחשף בפנינו הרשימה של כל המשתמשים החסומים:



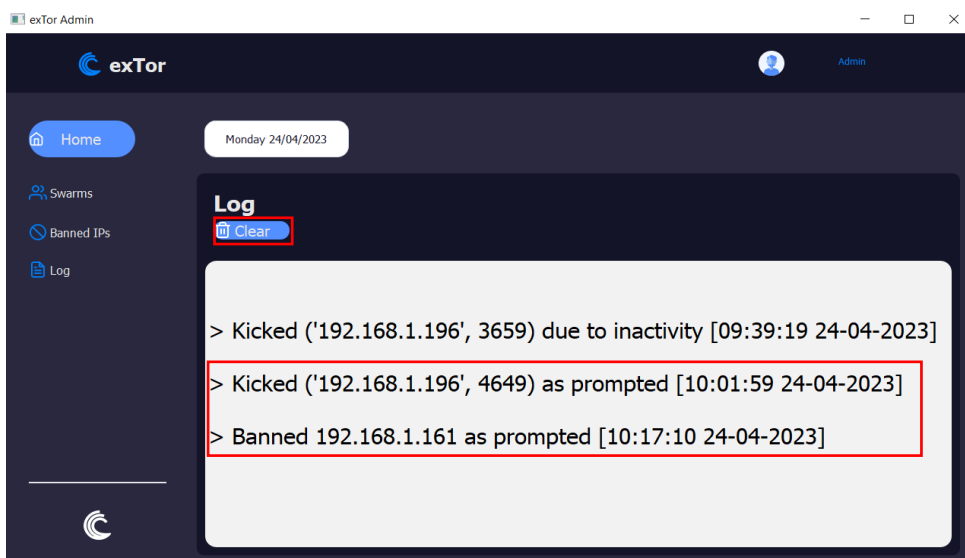
במקרה זה ניתן לראות (כמסומן באדום) שה-IP 192.168.1.161 חסום, חסמתי את ה-IP הזה בדרך שהוצגה לעיל.

באפשרותנו לבטל את החסימה על ידי לחיצה על הכפתור Remove, במקרה זה אעשה זאת.



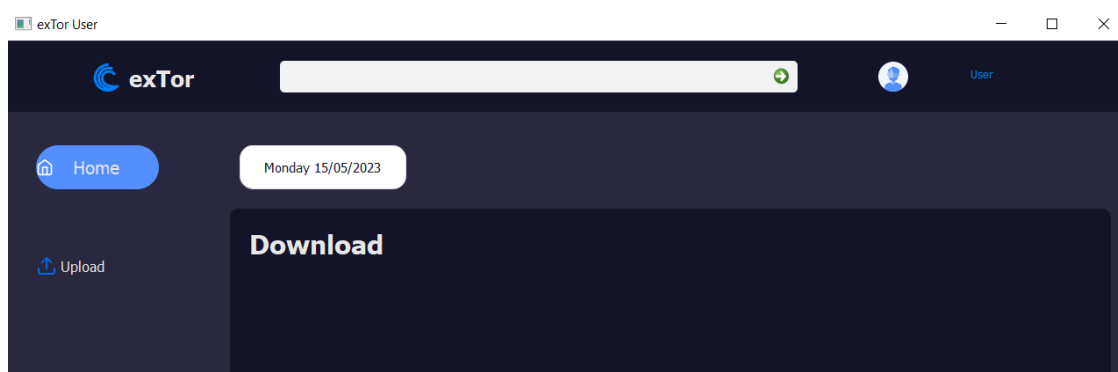
5. חלון Log:

השורה הבאה בתפריט השמאלי היא Log, במידה ונלחץ על כפתור זה יוצג בפנינו התיעוד של ה-Admin המחובר (הפעולות החשובות שביצע):

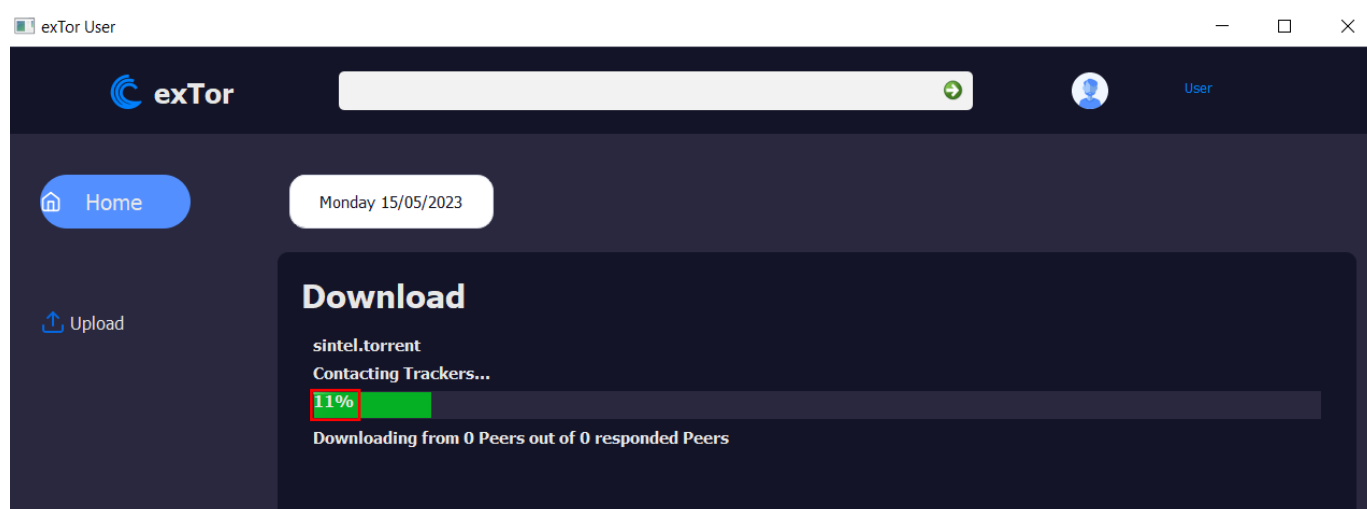


ב-2 השורות האחרונות ניתן לראות את הפעולות האחרונות שביצענו. ניתן גם לנקות את ה-Log, באמצעות לחיצה על הכפתור Clear.

1. חלון ראשי (חלון ההורדה)



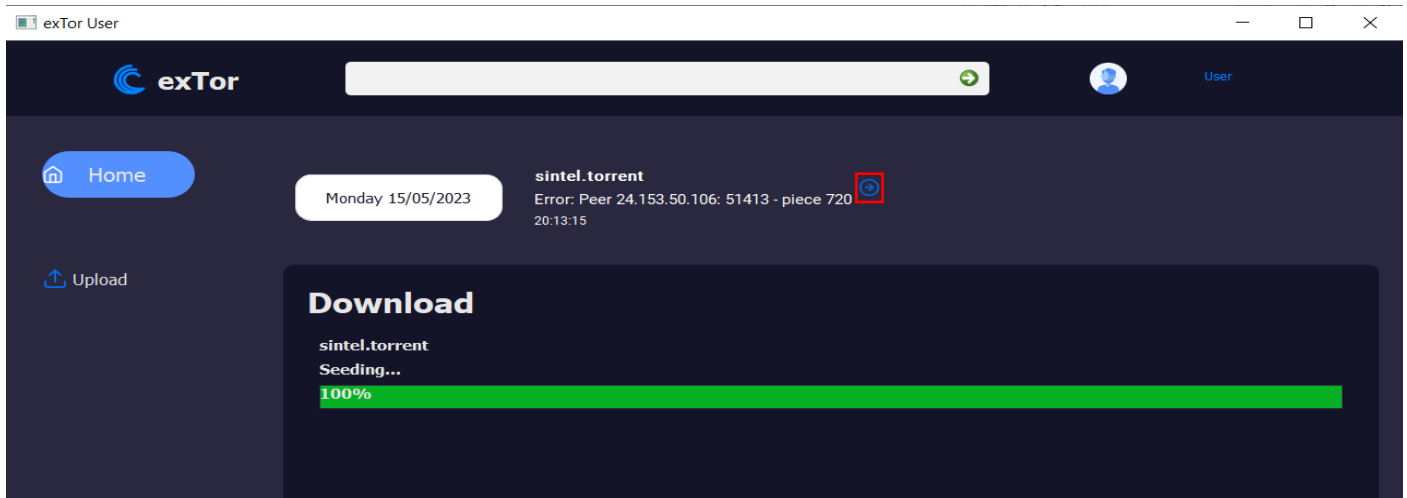
חלון זה הוא "מנהל ההורדות" שלנו, כל קובץ אותו נחפש או נעלה יופיע כרשומה בחלון זה, לדוגמה: נחפש בשורת החיפוש למעלה את הקובץ "sintel" לצורך הורדת הקובץ:



כפי שניתן לראות באדום, נמצא על פי אלגוריתם ש-11% מה-pieces של קובץ המטרה כבר נמצאים במחשב, הפעולה אותה מבצע כרגע כתובה מעל ה-progress bar, שהיא פנייה ל-Trackers שנמצאים בקובץ הגלובלי שהורד (לצורך השגת swarms), פעולות אשר יכולות להיות כתובות שם כוללות:

- Contacting Trackers... - פנייה לשרתי ניהול בקובץ
- Mapping pieces held by peers... – מיפוי העמיתים לחלקים שברשותם
- Downloading and Seeding... – מוריד חלקים (ומשתף)
- Seeding... – משתף חלקים

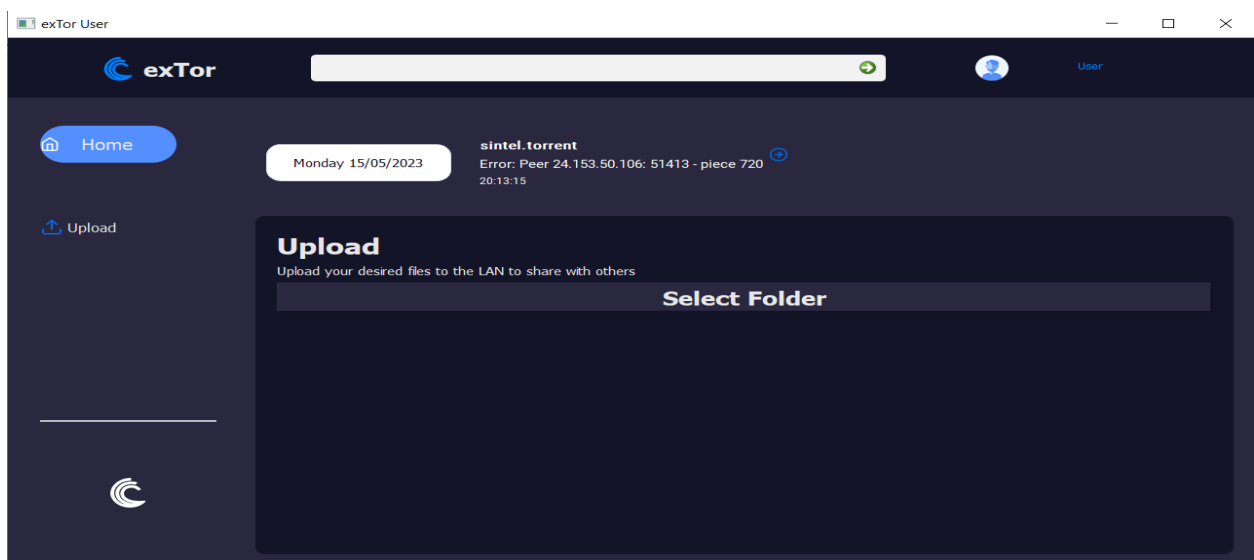
במהלך ההורדה אנחנו עלולים להיתקל בשגיאות בהורדות החלקים, על מנת להמחיש שגיאות אלה קיימות התראות שיוספו במידה ונתקל בשגיאה בהורדת אחד מה-pieces:



אנחנו יכולים לגלול בהתראות אלה על ידי לחיצה על הכפתור שליד ההתראה (באדום), לחיצה עליו תציג את ההתראה הבאה, חשוב לציין כי כל התראות השגיאות שמוצגות בחלון זה טופלו או נמצאות בטיפול על ידי אלגוריתם בקרת השגיאות, הן נועדו להמחיש את ההורדה טוב יותר.

2. חלון העלאת קבצים

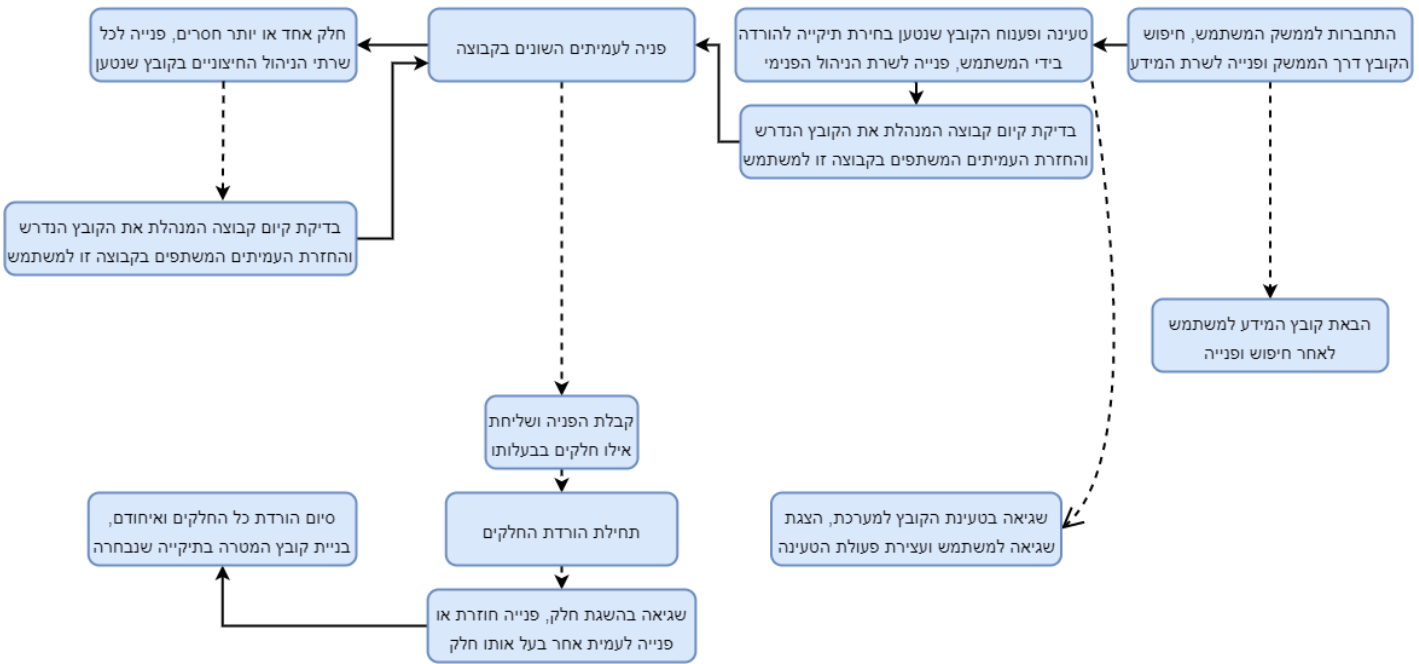
חלון זה הוא פשוט, דרך חלון זה ניתן לבחור תיקייה מהמחשב אותה נרצה להעלות



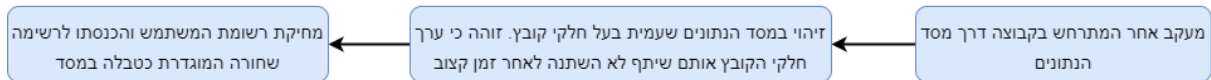
לאחר בחירה אלגוריתם העלאת הקבצים יפעל אוטומטית ויעלה את הקובץ, כאמור, לאחר העלאת הקובץ תתווסף רשומה בחלון הראשי עם המצב "Seeding" - שיתוף הקובץ.

3.7. תרחישים עיקריים

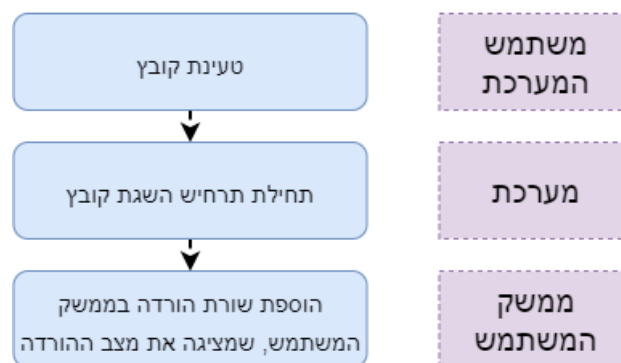
תרחיש השגת קובץ

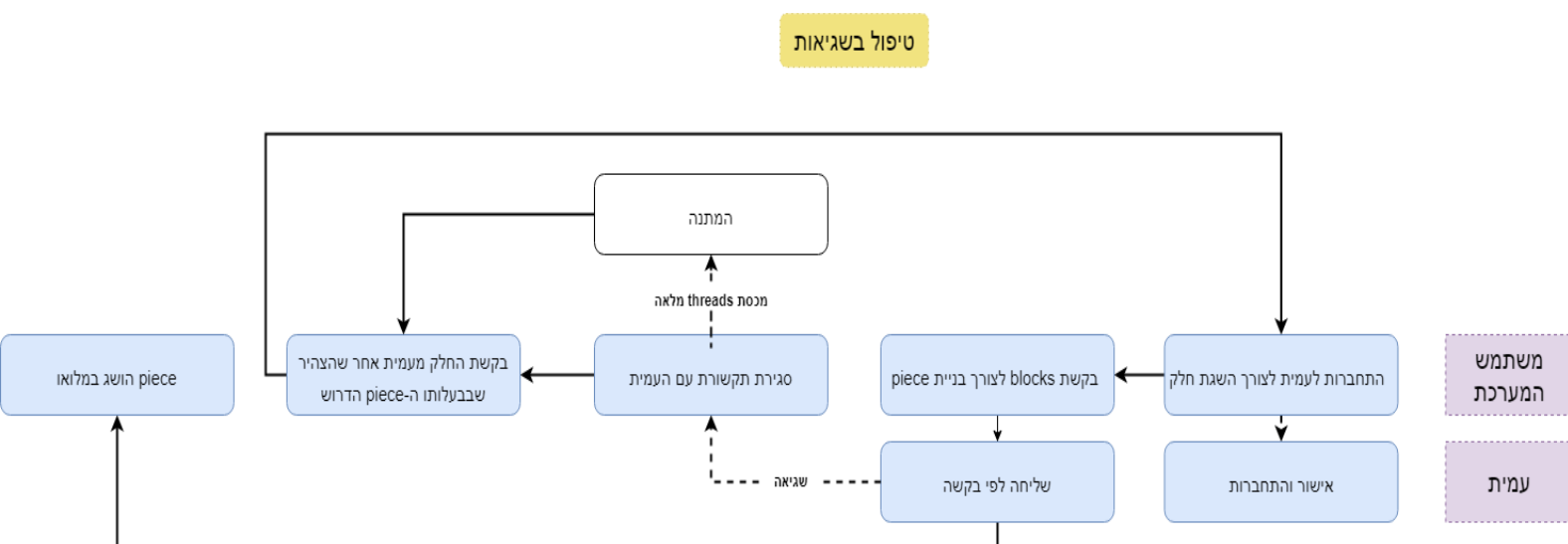


תרחיש חסימת משתמשים (admin)



טעינה מקבילית





4. תהליך כתיבת הפרויקט

4.1. תהליך הפרויקט

את הרעיון לכתיבת הפרויקט קיבלתי כבר בקיץ בשנה שעברה, תחילה רציתי שהפרויקט יהיה ממשק BitTorrent כדוגמת qBitTorrent אך לאחר מחקר מעמיק גיליתי שישנו צורך במערכת לשיתוף קבצים P2P ברשת הפנימית ולא רק בחיצונית, לכן החלטתי לשנות את הפרויקט ו"לשדרג" אותו. התחלתי לעבוד על הפרויקט לאחר שסיכמנו אני והמנחה על אופי הפרויקט וכל הרכיבים הקיימים בו, בהתחלה לא הייתי בקיא בידיעת הפרוטוקול BitTorrent, אך לאט לאט נחשפתי לפרוטוקול תוך למידה עצמאית שלו דרך ה-BEPs של הפרוטוקול ובעזרת אתרים שונים שמרחיבים את הדוקומנטציה הנתונה.

לקחתי על עצמי אתגר די גדול בבחירה של Python כשפת התכנות של הפרויקט, השפה אומנם מצוינת ב-threads שהיא מציעה אך ה-threads האלו איטיים בהשוואה לשפות אחרות כדוגמת C#. ב-POC שהצעתי בניתי מערכת שיתוף קבצים של הפרוטוקול BitTorrent ללא הורדה מקבילית, האתגר העיקרי כאן היה ללמוד את הפרוטוקול לעומקו ולאפשר טיפול ברוב התרחישים. לאחר מכן שדרגתי את המשתמש, בכך שאפשרתי לו לבצע חיפוש של קבצי torrent. בצד שרת הניהול, והתאמתי את שרת הניהול לאופן שבו הוא פועל כמוגדר בפרוטוקול BitTorrent. מאוחר יותר מימשתי הורדה מקבילית בצד המשתמש, ואפשרתי לו לפנות לרשת החיצונית במידה והוא לא מצא piece ברשת הפנימית, בשלב זה התחלתי לעבוד על בקרת שגיאות בצד המשתמש. בשלב בשלישי שדרגתי את שרת הניהול על ידי הוספת Admin כרכיב חדש במערכת, והתחלתי לבנות GUI לרכיב זה. הצעתי תכונות נוספות לשרת הניהול ולמשתמש.

בשלב האחרון כתבתי את ממשק המשתמש, אפשרתי למשתמש לחפש בממשק קובץ תוכן והשגתי אותו מהרשת בהתאם, הוספתי תמיכה בהורדה מקבילית של מספר קבצי תוכן. למדתי המון על כתיבת פרויקט ועל הקשיים הרבים הטמונים בו, אני מאמין שהרחבתי את הידע שלי רבות והרווחתי מיומנויות חדשות בזכות הפרויקט הזה, על אף הקשיים שבו.

4.2. אתגרים ואופציות שונות למימוש

- בהתחלה נתקלתי בקושי במימוש התקשורת המקבילית בצד משתמש המערכת, הבעיה העיקרית הייתה החוסר באלגוריתמים קיימים באינטרנט כך שהייתי צריך לפתור כל אספקט, וזה לקח לי יותר זמן משתכננתי אך לבסוף הגעתי לתוצאה מרצה, עם אלגוריתם ייחודי.
- תהליך כתיבת ה-GUI היה מאתגר משום שלא הייתי בקיא בספריה PyQT וחסר ניסיון בה, אך למדתי את הספריה לעומקה ובזכות זאת הצלחתי לבנות GUI ל-Admin ולמשתמש
- כשמימשתי את אלגוריתם קישור העמיתים ל-pieces שברשותם עשיתי זאת בצורה אסינכרונית, לאחר בדיקות ומחשבה עמוקה החלטתי לשכתב את האלגוריתם לצורה מקבילית עם threads, בחירה שהאיצה את תהליך הבירור ועזרה לי לבסוף גם במימוש ה-threads בצד משתמש המערכת והעמיתים שברשת.

5. מרכיבי פתרון

5.1. תיחום הפרויקט

- תקשורת – שרת, לקוח ברמה 4 של מודל ה-OSI, וברמה 7 על מנת לבצע תקשורת בין משתמש לשרת הניהול ברמת האפליקציה
- מערכות הפעלה – שימוש נרחב ועיקרי ב-threads להשגתו ולשיתופו של המידע אצל המשתמש
- תצוגה – שימוש בספריה PyQt לבניית ה-GUI של ה-client, וגם ב-pyqtgraph לתצוגה גרפית של admin
- אבטחה – שם משתמש וסיסמה ב-admin, מסד נתונים המשותף בינו לבין שרת הניהול בעל קוד אבטחה, SSL Sockets ברשת הפנימית
- מבנה נתונים – שימוש נרחב במערכים וב-Dictionaries כחלק מפענוח קובץ ה-torrent, שמירת מידע על עמיתים והחלקים שברשותם אצל המשתמש, מעקב אחר ה-pieces שהורדו בצד המשתמש, ועדכון ובדיקת מידע בזמן קצוב מראש אצל שרת הניהול לגבי העמיתים המחוברים לקבוצה.
- שימוש במסד נתונים משותף בין ה-admin ושרת הניהול

- ארכיטקטורת קוד – שימוש רב במחלקות ובפונקציות, פיצול הקבצים למספר רב של קבצים על מנת לאפשר טיפול באספקטים שונים לדוגמת פיצוח ההודעה, תקשורת עם tracker, תקשורת עם עמית, העלאת קובץ torrent. ל-tracker וכו'.
- תיעוד – תיעוד מלא של הקוד, כל פונקציה עיקרית מקבלת הסבר מפורט לגבי המשתנים ומטרתם, במידה ונעשה יישום של אלגוריתם בדרך שונה, אך בחרתי בדרך אחרת מזו, אשאיר את האלגוריתם בקוד ב-region, לדוגמה יישום האלגוריתם קישור בין עמיתים וה-pieces שברשותם באמצעות threads ויישום שונה בצורה אסינכרונית באמצעות הספרייה .asyncio

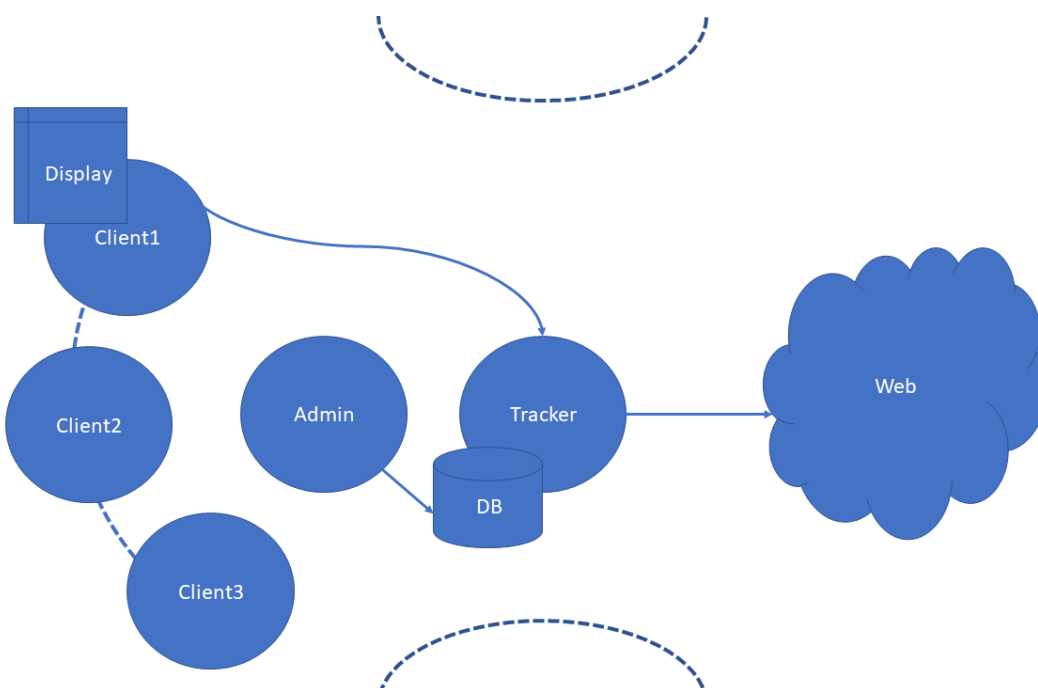
5.2. סביבת העבודה (טכנולוגיה)

- שפות התכנות:
המערכת ברובה כתובה ב-Python: שרת, לקוח, admin, שרת ניהול, GUI השימוש ב-threads ב-Python הוא נוח מאוד. בגלל השימוש המסיבי של threads בפרויקט בחרתי ב-Python על אף שהשפה יכולה לעיתים להיות איטית יחסית. נעשה שימוש גם ב-SQL למסד הנתונים הפרטי של שרת הניהול (זה שלא משותף לו ול-admin).

- סביבת פיתוח:
PyCharm – לשפת Python.

5.3. מבט טופולוגי

המערכת מורכבת משרת ראשי, מ-Admin ומ-Clients.



Client1, Client2, Client3 מייצגים עמיתים ברשת הפנימית, שמשתפים pieces זה עם זה. Client1 ראה שחסר לו piece ושהוא איננו נמצא ברשת הפנימית, לכן הוא פונה לרשת החיצונית לצורך השגת ה-piece החסר. כל העת ה-Admin ניגש למסד הנתונים המשותף לו ולשרת הניהול ומעדכן את הסטטיסטיקות.

5.4. [מבנה נתונים](#)

:Tracker

- admin_ips – רשימה של כל ה-admins המחוברים
- not_listening – רשימה של העמיתים אליהם לא יאזין שרת הניהול, עמיתים יוספו למבנה נתונים זה כשהם מעלים קובץ לשרת הניהול ויוסרו בסיום ההעלאה
- requests – רשימה של כמות הפניות לשרת הניהול ומילון של המשתמשים שפנו וכמה פניות ביצעו

:User

- Pieces – מילון של ה-pieces ואילו עמיתים מחזיקים ב-pieces אלו
- peer_list – רשימה של האובייקטים של העמיתים השונים (לצורך הורדה מהם)
- peer_thread – מילון של ה-threads הפתוחים ואובייקט העמית אליו ממנו מורד piece ב-thread זה
- currently_connected – עמיתים אליהם מחובר משתמש המערכת בעת ההורדה
- peers – רשימה של כל העמיתים שהושגו מהשרת הניהול השונים ברשת
- announce_list – שרתי הניהול הזמינים (במקרה של קובץ גלובלי) או העמיתים הזמינים (במקרה של קובץ מקומי)

:Admin

- ip_files – רשימה שמכילה בתוכה את כל העמיתים שלא שיתפו בטווח זמן נתון מרגע שנוספו, שולחת לשרת הניהול הודעה להסרתם ומאפסת את הרשימה

5.5. [מסד נתונים](#)

קיימים 2 מסדי נתונים, אחד של Redis והשני בשפת SQL שמנוהל על ידי מודל sqlite3, שניהם נמצאים על המערכת של שרת הניהול. המסד של Redis הוא מסד הנתונים המשותף בין שרת הניהול ל-Admin, זה של SQL הוא של שרת הניהול בלבד וכולל את שמות המשתמש והסיסמאות של ה-Admins.

New Database	Open Database
Database Structure	Browse Data
Table: Admins	
user	password
Filter	Filter
1	admin

במסד SQL: קיים Table אחד בשם Admins שכולל בתוכו את שם המשתמש והסיסמה של כל Admin.

מוד Redis:

במסד זה ישנו מילון של שם הקובץ והעמיתים שבתוכו, ומילון של כל עמית והשעה בה נוסף.

Swarms	
Group of Peers per local file	
File Name	
1 TestFile_UPLOAD.torrent	

Swarms	
Group of Peers per local file	
IP:PORT	Time Added
1 192.168.1.196:4649	Mon Apr 24 12:59:40 2023

5.6. מבט מודולרי

החלקים המרכזיים בפרויקט:

- Tracker – השרת המרכזי בפרויקט, תפקידו הוא לנהל את המערכת (הקבוצות השונות, לבצע ניטור), הוא מפיק סטטיסטיקות של כמות הפניות אליו. הקבוצות השונות נוצרות על ידו ועל המערכת שלו נמצא מסד נתונים משותף בינו לבין ה-Admin, אותו הוא מעדכן בהתאם למתרחש, ה-Tracker מורץ כקובץ Python.
- User – משתמש מערכת שנעזר בשרת הניהול ו-Peers אחרים לצורך השגת קובץ רצוי על ידו, ה-User הוא חלק מהמערכת הסגורה באופן בו איננו יכול לפנות לרשת החיצונית ללא עזרתו של שרת הניהול. ה-Peer משתמש ב-threads לצורך הורדת pieces של קובץ רצוי על ידו באמצעות ביצוע תקשורת מקבילית. ה-User מורץ כקובץ Python.
- Admin – כלי עזר של Tracker, תפקידו העיקרי הוא להוריד מעומס ה-Tracker ותפקידו המשני הוא להציג בתצוגה גרפית את הסטטיסטיקות שמופקות על ידי ה-Tracker. הוא מתחבר ל-Tracker בעזרת socket ומורץ כקובץ Python.

5.7. פירוט מודלים עיקריים

Class	Function	Input\Output	Description
Tracker	init_udp_sock	Input: int Output: obj	יוצר UDP socket בעל port נתון

	listen_udp	Input: None Output: None	מאזין להודעות UDP משתמשים ומטפל בכל הודעה בהתאם לסוגה
	add_peer_to_LOC	Input: string, tuple Output: None	מוסיף עמית נתון לקובץ torrent מקומי ולמסד הנתונים
	send_files	Input: string, string, tuple Output: None	שולח קובץ torrent. גלובלי וקובץ torrent. מקומי (אם קיים) למשתמש נתון
	torrent_from_web	Input: string, tuple, obj Output: None	מחפש שאילתה נתונה ברשת החיצונית ומוריד קובץ torrent. גלובלי במידה ונמצא כזה
	send_torrent_file	Input: string, tuple, bool Output: None	שולח קובץ torrent. למשתמש נתון
	init_tcp_sock	Input: None Output: obj	יוצר TCP socket בעל פורט קבוע
	listen_tcp	Input: None Output: None	מאזין להודעות TCP משתמשים ומטפל בכל הודעה בהתאם לסוגה
	recv_files	Input: obj, string Output: None	מוריד קובץ metadata שהועלה בידי משתמש מערכת
	ban_ip	Input: string, obj Output: None	חוסם משתמש נתון מהתקשרות נוספת עם שרת הניהול
User	create_new_sock	Input: None Output: obj	יוצר TCP socket בעל פורט רנדומלי
	Download	Input: None Output: None	מבצע סדרת פעולות שסופן יהיה השגת קובץ התוכן הדרוש או לחילופין חוסר יכולת לעשות זאת
	go_over_pieces	Input: None Output: None	עובר על pieces מהנדיר לנפוץ ומקצה עמיתים ל-pieces שאינם בבעלות המשתמש

	check_errors	Input: None Output: None	פעולה רקורסיבית שמתבצעת עד שאין עוד שגיאות בהורדת אחד מה-pieces הקודמים, מקצה עמית לכל piece שנתקל בשגיאה במקביליות
	peer_piece_assignment	Input: obj, int, list Output: None	עובר על כל העמיתים המחזיקים piece נתון ומנסה להקצות אותו לעמיתים במטרה להוריד את אותו ה-piece, עושה זאת במקביליות תוך בקרת שגיאות
	connect_to_peer	Input: list, obj, int, obj Output: None	שולח הודעה לעמית נתון ומפענח את הודעת ה-Bitfield שהוחזרה ממנו, מוסיף לרשימה משותפת את כל ה-pieces שבבעלותו
	rarest_piece	Input: list, obj Output: None	מוצא באמצעות אלגוריתם את כל ה-pieces הזמינים ברשת הפנימית או החיצונית והנדירות שלהם
	is_handshake	Input: string Output: bool	מזהה האם ההודעה היא handshake
	msg_type	Input: string Output: string	מזהה את סוג ההודעה ומחזיר אותה
	build_[type]	Input: obj Output: bytes	רצף של 11 פונקציות, תפקיד כל אחת מהן הוא לבנות הודעה מסוג מסויים של הפרוטוקול BitTorrent
	is_handshake_hash	Input: bytes Output: bool	בודק האם משתמש המערכת יכול לשתף hash נתון (אותו קיבל מעמית לצורך הורדה)
	request_piece	Input: int Output: None	שולח בקשת "request" לעמית כשמתפנה אחד כזה, במידה

			ועמית לא פנוי מחכה כשיתפנה ואז עושה זאת
	listen_to_server	Input: None Output: None	מאזין להודעות מעמיתים ברשת ושולח אותן לפענוח (לצורך הורדה בלבד ולא העלאה)
	message_handler	Input: bytes Output: None	מפענח הודעות שהתקבלו מעמיתים באופן שבו הן מוגדרות בפרוטוקול BitTorrent, ומבצע את הפעולות הדרושות בהתאם
	update_have	Input: None Output: None	שולח הודעת "have" לעמיתים ברשת המעדכנות את ה-pieces שבבעלותו
	calculate_bitfield	Input: None Output: None	מחשב את ה-pieces שבבעלותו משתמש המערכת באמצעות אלגוריתם בנוגע לקובץ שנבחר, כך המערכת יודעת אילו pieces להוריד במקרה של נפילה
	listen_seq	Input: None Output: None	מבצע 2 פעולות: מאזין להודעות מעמיתים ברשת שרוצים להוריד pieces שבבעלותו משתמש המערכת ומאזין להודעות משרת הניהול הפנימי
	add_piece_data	Input: int, bytes Output: None	מקבל את מספר ה-piece ותוכנו ושם את התוכן במקום המתאים בקובץ התוכן הסופי, עם מספיק תכני piece יוכל להרכיב את קובץ התוכן

	send_piece	Input: bytes, obj Output: None	שולח piece נתון לעמית נתון ברשת לפי הפרוטוקול BitTorrent
	calculate_file_piece	Input: None Output: None	ממפה כל piece למקומו בדיסק (יעיל ביותר במקרה של מספר קבצי תוכן)
	file_names	Input: None Output: None	מטפל בתוכנו של קובץ ה- metadata (torrent), במקרה של קובץ פנימי בכל מיקומו של tracker יהיה עמית אחר ברשת הפנימית
	contact_trackers	Input: None Output: None	פונה לכל שרת ניהול ברשת החיצונית בהתאם לסוג (UDP/TCP) לצורך השגת עמיתים
	listen (tracker.py)	Input: obj Output: None	האזנה ופענוח של הודעות משרתי ניהול ברשת החיצונית
	fetch_torrent_file	Input: None Output: string	משיג קובץ metadata (torrent) משרת הניהול הפנימי בהתאם לשאילתה
	find_local_tracker	Input: None Output: string	מוצא את שרת הניהול הפנימי באמצעות broadcast
	create_metadata_file	Input: string Output: string	יוצר קובץ metadata (torrent) מקובץ שברצון המשתמש להעלות
	listen (upload_file.py)	Input: None Output: None	האזנה לשרת הניהול הפנימי לצורך העלאת קובץ תוכן אליו
Admin	deleter_timer	Input: None Output: None	לולאה אינסופית, "מעיפה" עמיתים שלא משתפים בזמן קצוב מאז שנוספו
	update_widgets	Input: None Output: None	מעדכן את הגרף וה-log בכל 5 שניות

6. תסריטי בדיקה

6.1. [דגשים בבדיקה](#)

- המערכת יודעת להמשיך להוריד pieces מהנקודה בה המשתמש עצר
- המערכת פונה לרשת החיצונית במקרה של מחסור ב-pieces ברשת הפנימית
- המערכת שומרת קבצי תוכן בדיסק נכונה
- הסטטיסטיקות המופקות בידי שרת הניהול נכונות ומדויקות
- המערכת יודעת לטפל בשגיאות תקשורת בעת הורדת pieces
- המערכת פונה לכל שרתי הניהול החיצוניים בקובץ metadata (.torrent) ומשיגה מהם קבוצות
- קובץ שהועלה בידי משתמש מערכת יכול להיות מורד בידי כל משתמש מערכת אחר ברשת הפנימית

6.2. [תסריטי בדיקה עיקריים](#)

- חיבור 3 משתמשים, Admin ושרת ניהול על מערכות שונות
- חיבור Admin לשרת הניהול לצורך תצוגה גרפית
- חיבור משתמש 1 לשרת הניהול להורדה, הורדת קובץ תוכן על ידו מעמיתים ברשת החיצונית
- לאחר סיום ההורדה של משתמש 1 לחבר משתמש 2 לשרת הניהול, ולחפש את אותה השאילתה אותה שלח משתמש 1, להוכיח שההורדה על ידו תיעשה ממשתמש 1 ולא מהרשת החיצונית
- לנתק את משתמש 2 בזמן הורדת החלקים ממשתמש 1, לחבר את משתמש 2 לשרת הניהול ולחפש את אותה השאילתה, להוכיח שההורדה ממשיכה מהנקודה בה פסקה
- לחבר משתמש 3 לשרת הניהול ולהוריד חלקים ממשתמשים 1 ו-2, להראות שההורדה מקבילית
- לנתק את כלל המשתמשים, לחבר את משתמש 1 לשרת הניהול ולהעלות דרכו קובץ שנמצא במחשב שלו, לחבר את משתמש 2 ולהוריד את הקובץ שהועלה על ידו
- לחבר את משתמש 3 ולהוריד את הקובץ שהועלה על ידי משתמש 1, להראות שההורדה מקבילית ממשתמשים 1 ו-2
- לעבור ב-Admin על המסד המשותף בינו לבין שרת הניהול, להראות את הקבוצות שנוספו (2 לפי התסריט), "להעיף" ולחסום משתמשים, לחבר את המשתמשים שנחסמו ולהוכיח שהחסימה התבצעה

7. רפלקציה

7.1. לוח זמנים מוערך לניהול הפרויקט:

נובמבר- דצמבר	תחילת כתיבת פרוטוקול הפרויקט, כתיבה של כחצי מיישום פרוטוקול BitTorrent תוך תמיכה בכל התכונות שלו – השגת ה-peers מ-tracker, להתחבר כ-peer, לנתח קובץ torrent, הורדת קבצים מ-peers אחרים, תמיכה בהעלאה, טיפול בשגיאות.
ינואר- פברואר	סיום כתיבת יישום פרוטוקול BitTorrent, תמיכה בהעלאה, הורדה, טיפול בשגיאות אפשריות, תחילת העבודה על שרת ניהול, קובץ מידע, פעולות בין משתמשים, תחילת העבודה על ממשק המשתמש.
מרץ- אפריל	סיום כתיבת פרוטוקול הפרויקט (למעט שרת ניהול), סיום העבודה על שרת הניהול*, תחילת העבודה על ממשק ה-admin, המשך עבודה על ממשק המשתמש וטיפול בבאגים.
מאי-	סיום כתיבת ממשק המשתמש, סיום כתיבת ממשק ה-admin. בדיקה סופית של הפרויקט עם כל האלמנטים שבו.

7.2. אתגרים ותרומה אישית

הפרויקט הזה הוא התוצר הגדול ביותר שביצעתי בחיים שלי. בתחילת השנה עוד לא הייתי סגור על אופיו של הפרויקט ובזכות הסכמות עם המנחה הצלחתי להגדיר את הפרויקט באופן בכללותו. כתיבת ספר זה הקלה מאוד על כתיבת הפרויקט, היא סידרה לי את היישויות והקשרים השונים ובזכות זאת הצלחתי להפוך את הפרויקט לדבר אמיתי.

האתגר העיקרי בפרויקט זה הוא הלמידה של חומר חדש, נדרש ממך למצוא הרבה זמן פנוי לצורך למידה, והעומס הלימודי לא הקל בכך, עמדתי באתגר הלחץ ולמדתי לנהל את הזמן שלי יותר טוב.

הפרויקט העסיק אותי רבות בתקופה הזאת, היו צדים רגעים בהם פתאום עלה בי רעיון יצירתי ואת רוב הרעיונות האלו הצלחתי לממש כחלק מהפרויקט.

7.3. תובנות

כתיבת הפרויקט הייתה קשה ומאתגרת במיוחד. בהתחלה הייתה בי מוטיבציה רבה אך עם הזמן כאשר הבנתי את גודלו של הפרויקט ירדה לי המוטיבציה. היו הרבה יאושים ותסכולים, לדוגמה כשבמחשבים בבית הספר לא עבדה תקשורת עם הרשת החיצונית בגלל חסימות.

הצלחתי להתגבר על הקושי שהפריקט הסב לי משום שהצלחתי לסדר את הזמן בצורה הנכונה ומצאתי פתרונות יצירתיים לבעיות שנראו לי תחילה לא פתירות כלל, לאורך כל הדרך לא התייאשתי גם כשהיה קשה ואני גאה על כך, למדתי המון והתפתחתי בתקופה הזאת, הפרויקט גרם לי להעריך שלעבודה קשה יש משמעות.

8. הוראות התקנה ותפעול

8.1. תצורה ודרישות קדם

למחשב ה-Tracker:

- נדרשים ספריות: threading, time, socket, sys, os, pickle, requests, select, diffli, py1337x, bencode, urllib3, sqlite3, redis, random, ssl
- Python גרסה 3.4 ומעלה

למחשב ה-User:

- נדרשות ספריות: os, socket, threading, time, datetime, bitstring, atexit, struct, hashlib, pickle, shutil, PyQt5, urllib, random, bencode, requests, ssl, torf, warnings, sys
- Python, גרסה מומלצת: 3.9

למחשב ה-Admin:

- נדרשות ספריות: PyQt5, sys, datetime, warnings, time, redis, numpy, threading, pickle, socket, ssl, pyqtgraph, math, os,
- Python, גרסה מומלצת: 3.9

8.2. התקנה

תחילה יש לחבר את שרת הניהול, לאחר מכן מומלץ לחבר את ה-Admin ולאחר מכן את משתמשי המערכת, מכאן התוכנה תיתן הוראות בעצמה.

9. ביבליוגרפיה

במהלך כתיבת הפרויקט הסתמכתי על מספר מקורות מידע:

[/https://stackoverflow.com](https://stackoverflow.com)

<https://wiki.theory.org/BitTorrentSpecification>

[/http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1](http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1)

[/http://www.kristenwidman.com/blog/71/how-to-write-a-bittorrent-client-part-2](http://www.kristenwidman.com/blog/71/how-to-write-a-bittorrent-client-part-2)

<https://www.bittorrent.org/beps>

https://en.wikipedia.org/wiki/Torrent_file

10. נספחים

הצעת פרויקט:

<https://docs.google.com/document/d/1txVlemm5O9JM0Otnn7YjrDLOZcpLtqPVfHg6P-BikRY/edit?usp=sharing>

מבנה הקובץ .torrent:

- Announce — the URL of the tracker
- Info — this maps to a dictionary whose keys are dependent on whether one or more files are being shared:
 - Files — a list of dictionaries each corresponding to a file (only when multiple files are being shared). Each dictionary has the following keys:
 - length—size of the file in bytes.
 - path—a list of strings corresponding to subdirectory names, the last of which is the actual file name
 - length — size of the file in bytes (only when one file is being shared though)
 - name — suggested filename where the file is to be saved (if one file)/suggested directory name where the files are to be saved (if multiple files)
 - piece length — number of bytes per piece. This is commonly 2^8 KiB = 256 KiB = 262,144 B.
 - pieces — a hash list, i.e., a concatenation of each piece's SHA-1 hash. As SHA-1 returns a 160-bit hash, each piece will be a

string whose length is a multiple of 20 bytes. If the torrent contains multiple files, the pieces are formed by concatenating the files in the order they appear in the files-dictionary (i.e., all pieces in the torrent are the full piece length except for the last piece, which may be shorter).

נלקח מ: https://en.wikipedia.org/wiki/Torrent_file