

**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 :** 20/12/2022

תוכן עניינים

[1. מבוא 3](#_Toc121815660)

[2. תיאוריה 4](#_Toc121815661)

[3. תוצר סופי 10](#_Toc121815662)

[4. תהליך כתיבת הפרויקט 20](#_Toc121815663)

[5. מרכיבי פתרון 20](#_Toc121815664)

[6. תסריטי בדיקה 21](#_Toc121815665)

[7. רפלקציה 22](#_Toc121815666)

[8. הוראות התקנה ותפעול 22](#_Toc121815667)

[9. ביבליוגרפיה 23](#_Toc121815668)

[10. נספחים 23](#_Toc121815669)

1. מבוא
   1. נושא העבודה

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

בנוסף, מיושם ממשק גרפי בצד ה-Admin, כחלק ממשק זה, יכול ה-Admin לראות את כמות הפניות לשרת הניהול בכל עת, לעבור על ה-database המציג את מצב המשתמשים המשתפים בקבוצות שונות ברשת הפנימית, לראות את המשתמשים החסומים וגם לראות את ה-log המציג את הדברים העיקריים אשר עשה ב-session, 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. מטרות 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 כמות ההודעות שהשרת מקבל ממשתמשים בכל זמן קצוב, וגם סטטיסטיקות העוקבות אחר המשתמשים הנמצאים בכל קבוצה (Swarm), כחלק ממעקב זה יתאפשר לדעת האם משתמש שיתף חלקים או לא בפרק זמן מסוים, ל-admin תינתן גישה מלאה לכלים אלו ודרכו יוצגו גרפים העוקבים אחר המתרחש, ותתבצע מחיקה אוטומטית של משתמשים אשר לא שיתפו בקבוצה מסוימת.
* התאמת פתרון מיטבי לדרישות דינמיות של המשתמש, כולל טיפול בשגיאות תקשורת, וניצול מלא של תקשורת מקבילית במערכת, על מנת שתתאפשר הורדת חלקים והעלאתם ללא השהייה.

המטרות האישיות של העבודה הן:

* למידת פרוטוקול BitTorrent לכל רוחבו על מנת שאוכל ליישם את רוב הדברים הכתובים בו וגם 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. 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 השחקנים 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 מההיבטים 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 התקשורת ולהרחיב את הידע שלי ב-python, C++, ו- SQL.

* 1. 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 עם 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 מסיבי ב-threads כחלק מהחומר הנלמד בתחום מערכת ההפעלה, כדי לאפשר מקביליות, כמוזכר.
* 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 אצל ה-admin, תצוגת מצב הורדת הקובץ, וכו'.

1. תיאוריה
   1. תיאוריה

הפרויקט עוסק ברובו בתחום התקשורת, כחלק מהמערכת ייושמו:

בשלב זה אני קורא את התיאוריה רק כדי להבין טוב יותר מה אתה עושה. יש לכתוב את הפרק הזה אחרת כך שלא תהיה הפניה ממושג אחר למושג שעדיין לא הצגת קודם לכן, למנוע חזרות מיותרות, וכן לפתור את כל הקונפליקטים.

יש להחליף את המושג מנהל במושג admin.

* קובץ .torrent הינו קובץ מסוג Metadata, קובץ זה מספק את האינפורמציה הדרושה להורדת קובץ המטרה השלם, את קובץ זה משיג המשתמש משרתי מידע הנמצאים באינטרנט. המערכת חייבת לתמוך בקובץ זה על מנת שתתאפשר הורדת קובץ התוכן בידי משתמש המערכת, שכן קובץ זה מספק את המידע הנחוץ להורדה, בתוכו:
* כתובות שרתי הניהול (trackers) וסוגם (UDP/HTTP/WebTorrent) בנוסף ל-PORTS שלהם.
* גודל קובץ / קבצי התוכן, ושמם
* שם תיקיית הקובץ / קבצי התוכן
* גודל piece, לא כולל ה-piece האחרון
* רשימת ה-hashes של ה-pieces לצורך וידואם לאחר הורדה

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 28KiB = 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, pieces 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 filesdictionary (i.e., all pieces in the torrent are the full piece length except for the last piece, which may be shorter).

מאת: <https://en.wikipedia.org/wiki/Torrent_file>

* קבצי תוכן – כל קובץ תוכן שברצון משתמש להשיג, בנוי מ-pieces בעלי גודל קבוע שמצוין בקובץ .torrent, פרט ל-piece האחרון שגודלו אינו קבוע. כל piece בנוי מ-blocks. על מנת להשיג את הקובץ המיוחל, על העמית לבקש מהעמיתים השונים blocks עד שיהיו ברשותו מספיק blocks ל-piece, הוא עושה זאת עבור כל חלק וחלק עד שנבנה קובץ התוכן השלם.
* עמית (Peer) – עמית הוא חלק מ-swarm (קבוצה), עמית יכול להיות במספר קבוצות בו זמנית, זאת בהתאם למספר שרתי הניהול (Trackers) אליהם פנה. כל עמית משתף חלקי קובץ עם עמית אחר, ובכך יכול להרכיב קובץ תוכן שלם. עמית גם משתף חלקים וגם משיג אותם מהעמיתים האחרים ב-swarm, בהתאם לצרכיו. במידה ועמית סיים להוריד piece, הוא יעדכן את כל המשתמשים ב-swarm שברשותו חלק חדש, כך שכל העמיתים ידעו על מצב כל העמיתים בכל עת.
* שרת ניהול (Tracker) – שרת אשר מנהל קבוצות עמיתים שונות בהתאם לקבצי ה-.torrent שנמצאים לרשותו. שרת הניהול כמוגדר בפרוטוקול BitTorrent יכול לקבל קבצי .torrent ממשתמשים ובכך ליצור קבוצות עמיתים חדשות.

שרת הניהול יקבל את קבצי ה-.torrent ממשתמשים ויעדכן את מסד הנתונים על ידי יצירה של table חדש כאשר שמו יהיה ה-SHA1 hash של ה-info section של קובץ ה-.torrent שהתקבל מהמשתמש, כך שכאשר שרת הניהול ירצה להוסיף משתמש חדש ל-swarm יבדוק האם קיים table בשם ה-info hash שקיבל מהמשתמש, table זה צריך להישאר פעיל כל הזמן בשרת הניהול, גם לאחר מחיקת רשומת המשתמש שיצר את הקבוצה, אומנם, מנהל יכול למחוק את table זה כלומר את ה-swarm כחלק מהפונקציונליות שלו. את זה עדיין לא הבנתי! אתה מתכוון לקובץ swarm חדש? איפה רשומים כל הקבצים של ה swarm? כאשר משתמש מתחבר לשרת הניהול ומבצע את תהליך בירור העמיתים (המחוברים ל-swarm) לגבי קובץ התוכן אותו המשתמש מעוניין להוריד, שרת הניהול יוסיף את המשתמש ל-swarm, כלומר ייצור רשומה חדשה שלו ב-table המתאים במסד הנתונים, לאחר זמן מה, כאשר יזהה שרת הניהול שהמשתמש איננו משתף יותר pieces, ימחק את הרשומה שלו ממסד הנתונים. להבנתי רק אם קובץ המידע נמצא אצל אחד המשתמשים בקבוצה. אחרת לא מוסיף.... ולאחר זמן מה, לאחר שהצליח להרכיב את קובץ המטרה, ינתק אותו??

שרת הניהול יודע מי מחובר לקבוצה, ולרשותו סטטיסטיקות, ובניהן גרפים וכלי אוטומטי לחסימת משתמשים אשר שולחים לו הודעות רבות ללא תכלית.

* Swarm – קבוצה אשר נוצרת על ידי שרת הניהול, התנאים להיווצרותה של קבוצה זו הם העלאה של קובץ בידי משתמש המערכת לשרת הניהול או היוועדות של שרת הניהול בכך שמשתמש מערכת סיים להוריד קובץ גלובלי אשר ניתן על ידו (בסיום הורדה של קובץ גלובלי משרת הניהול המשתמש יעדכן את שרת הניהול כשסיים להוריד קובץ זה).

swarm נוצר במסד הנתונים המשותף לשרת הניהול ול-admin כטבלה במסד, הרשומות בטבלה זו הם ה-IP והשעה בה נוספה הרשומה של כל עמית הנמצא ב-swarm זה. הקבוצה בנויה מהעמיתים שבה כך שנוצרת טופולוגית full mesh (כל העמיתים יודעים על מצב כל העמיתים האחרים בכל עת).

שרת ניהול יכול לנהל מספר swarms, כאשר כל swarm הוא table במסד הנתונים של שרת הניהול, וישנה הגבלה על מספר ה-swarms אותם שרת הניהול יכול לנהל, כלומר הגבלה על מספר ה-tables במסד הנתונים כדי לשמור על יציבותו.

* Database – מסד נתונים משותף בין שרת הניהול ל-admin, הנמצא במערכת בה נמצא שרת הניהול. במסד נתונים זה מתבצע עדכון של כל המתרחש ב-swarm עליו אחראי שרת הניהול, ובו נמצא ה-admin, במסד נתונים זה נשמר המידע על כל המשתמשים הנמצאים ב-swarm בכל רגע, כך שלדוגמה במידה ומשתמש פונה לשרת הניהול על מנת להצטרף ל-swarm מסוים, משיג שרת הניהול את כל הרשומות של המשתמשים הנמצאים ב-swarm הנדרש, שולח לו את ה-IP וה-PORT של משתמשים האלה, ומוסיף רשומה של המשתמש למסד הנתונים כעמית נוסף הנמצא ב-swarm.

כחלק מערכי רשומה במסד נתונים זה, נכללים ה-IP, ה-PORT של העמית, ה-pieces אשר העמית שיתף עם עמיתים אחרים, האם העמית מנהל, ומה הם החלקים שבבעלותו של העמית.

על מנת לגרום להיווצרות swarm חדש אצל שרת הניהול, זאת באמצעות היוודעו של קובץ .torrent חדש בידי שרת הניהול (לדוגמה קובץ .torrent שהועלה אליו בידי משתמש), ייווצר במסד הנתונים table חדש האחראי על נתוני המשתמשים ב-swarm של קובץ ה-.torrent שהועלה, כך נוצר swarm חדש בידי שרת הניהול.

* שרת מידע (Info Server) – שרת הנמצא ברשת הפנימית, באמצעות שרת זה, משיג המשתמש את קובץ ה-.torrent אותו דורש, לשרת מידע זה מלאי של קבצי .torrent שונים שאותם מספק למשתמש בהתאם לדרישתו.

לאחר שיחפש המשתמש בממשק המשתמש שם קובץ תוכן, יבדוק שרת זה האם קיים קובץ .torrent תואם, במידה וכן, ישלח אותו אליו, והמערכת תזהה את הקובץ שהתקבל ותטען אותו אוטומטית אליה, במידה ולא, ייעשה שימוש ב-x API1337 לצורך מציאת הקובץ מהרשת החיצונית והשגתו.

* ממשק משתמש – GUI של יישום הלקוח בפרוטוקול הפרויקט, ממשק המשתמש הוא הדרך שבה מיישם הפרויקט את הלקוח כפי שהוגדר בפרוטוקול.
* מנהל (admin) זוהי הגדרה חדשה בפרויקט זה, שכן הפרוטוקול BitTorrent לא מגדיר אותו, עם יכולות ניהול שונות המאפשרות לו לעקוב אחר המתרחש בקבוצה בה נמצא ולחסום עמיתים מ-swarms שונים בהתאם (לדוגמה: המנהל יזהה אוטומטית שליחה מופרזת של הודעות ויחסום את ה-IP מהתקשרות נוספת עם שרת הניהול).
  1. מוצרים קיימים

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

qBitTorrent – 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 לקוח BitTorrent בודד.

- thepiratebay הוא שרת הניהול המוכר והשמיש ביותר היום, עם 5,100,000 (!) עמיתים מחוברים בו זמנית, וניהול swarms של כ-630 אלף torrents שונים. השירות אינו קוד פתוח אך כולל סטטיסטיקות כמו רוב 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. אתה צריך לכתוב זאת מחדש על מנת להוריד את שגיאות WORD אדומות.

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 ממספר מובנים:

במרבית המוצרים הקיימים בשוק, כדוגמת BitTorrent, חסר מרכיב ה admin מנהל 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, איישם בפרוטוקול שלי גם admin, כחלק מהמערכת הסגורה, ברשת הפנימית, ואאפשר:

* לחסום משתמשים ספציפיים, בצורה ידנית.
* לנטר את מצבם של כל אחד מן המשתמשים ב-swarm המשותף – אילו pieces הורדו, אילו pieces שותפו עם עמיתים אחרים ב-swarm, וכו', אלגוריתם לחסימת משתמשים אצל המנהל ישתמש בנתונים אלו ויחסום את המשתמשים אשר לא משתפים pieces שבבעלותם מה-swarm – פחות מסכים לגבי החסימה האוטומטית, מסכים לגבי החסימה הידנית כמופיע לעיל.
* ליצור ולמחוק swarms על ידי יצירת או מחיקת tables במסד הנתונים המשותף בינו לבין שרת הניהול

בנוסף, אין כיום מערכת כמו זו המוצעת, שכוללת את כל הפונקציונליות של פרוטוקול BitTorrent מהעמית לשרת הניהול. 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:

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

1. תוצר סופי
   1. תיאור הפרויקט

להתחיל במילה פונקציונלית על המערכת כמו במבוא. עדיין חסר.

המערכת מאפשרת הורדה של קבצים לפי פרוטוקול BitTorrent מורחב, כך שתעבוד גם ב-LAN בפרוטוקול זה, וגם ברשת החיצונית באמצעות הפרוטוקול BitTorrent.

רכיבי המערכת העיקריים: שים לב לשינויים שעשיתי.

* Info Server (שרת המידע) – שרת באמצעותו ניתן להשיג את קובץ ה-torrent.
* Tracker (שרת הניהול) – מנהל קבוצות תקשורת כאשר תפקידו העיקרי להוסיף באופן אוטומטי משתמשים חדשים לקבוצה, שרת הניהול יודע על כל המתרחש באותה הקבוצה, מי הם המשתמשים באותה הקבוצה ולאילו משתמשים יש אילו חלקים, ולרשותו גם סטטיסטיקות של כמות הפניות לאותו השרת בכל שניה. כל התהליכים בשרת זה הם אוטומטיים. 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. תפקיד נוסף של שרת זה הוא קבלת קבצי .torrent ממשתמשים לצורך יצירת swarms חדשים דרך שימוש במסד הנתונים המשותף בינו לבין ה-admin.
* עמית (Peer או משתמש) – באמצעותו מתבצעת תקשורת עם עמיתים אחרים, כאשר תפקידו העיקרי הוא לעזור לאחרים בהתאם ליכולתו להשיג קובץ וגם לקבל עזרה מאחרים להורדת קובץ.
* admin – מנהל המהווה חלק משרשרת הקבוצה. ה admin יודע על כל המתרחש באותה הקבוצה באמצעות המידע אותו מספק לו שרת הניהול דרך מסד הנתונים המשותף, לרשות המנהל יכולות חסימת משתמשים ידנית, אפשרות למחיקת ויצירת tables במסד הנתונים, למה שיעשה זאת? ותפקידו הוא לוודא שהכול בקבוצה הולך כשורה, אם יזהה שעמית משתף באופן מועט עם עמיתים אחרים בקבוצה את החלקים שברשותו, יחסום אותו מהקבוצה, באופן אוטומטי, באמצעות אלגוריתם. איך עובד "יזהה" עם "אוטומטי"?

לרשותו גם סטטיסטיקות המנטרות את העמיתים בקבוצה וה-pieces שלרשותם, את המעקב ה-admin מבצע באמצעות מסד הנתונים המשותף בינו לבין שרת הניהול (Tracker), שמתעדכן כל הזמן. את זה כבר אמרת.

המערכת מיישמת Client-Server, בין משתמש שמעוניין להשיג את קובץ המידע (.torrent) לבין שרת המידע המספק לו את אותו הקובץ, וגם בין משתמש שרוצה להצטרף ל-swarm לבין שרת הניהול שמספק לו את המידע לגבי המשתמשים בקבוצה אליה מעוניין להצטרף, ומצרף אותו לאותה קבוצה. בנוסף, המערכת מיישמת תקשורתP2P , כאשר מספר עמיתים מחוברים ומתקשרים ביניהם בתוך הקבוצה בה הם נמצאים באופן מקבילי.

לאחר שמשתמש מעלה את ה-GUI , הוא יכתוב בשורת החיפוש מה ברצונו להוריד, המערכת תאתר קובץ .torrent תואם מהרשת (הפנימית או החיצונית במידה ולא נמצא בפנימית) ותטען אותו, בנוסף, המערכת תבצע פניה לשרת המידע ברשת הפנימית ותשאל אותו לגבי קיום של קובץ כזה אצלו. מה הכוונה בנוסך? הבנתי ששרת המידע מבצע את האיתור, לא? המערכת תבחן את קבצי ה-.torrent שהושגו ותוציא מהם את רשימת ה-trackers, תפנה אל כולם באמצעות אלגוריתם להשגת כל העמיתים מכל הקבוצות, ותתחיל בהורדה מהעמיתים השונים בהתאם ליכולתם, תוך השוואת כל piece מהחלקים שמרכיבים את הקובץ ל-hash המתאים בקובץ .torrent שנטען.

שרת הניהול (Tracker) ישמור את המידע לגבי כל משתמש וחלקיו הזמינים במסד הנתונים בצורה דינמית עם עדכונים כל זמן קצוב. שרת הניהול גם ימחק כל משתמש (חוץ מ-admin) אשר התחבר אליו, כלומר יסיר אותו מן הקבוצה לאחר שיזהה שמשתמש זה לא משתף חלקים עם עמיתים אחרים ב-swarm המשותף, שרת ניהול לעולם לא יסיר admin מקבוצה.

המערכת מאפשרת מספר פעולות:

* הורדת קובץ ממספר עמיתים כאשר כל עמית אחראי על pieces הנמצאים ברשותו והמרכיבים קובץ מטרה, 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-trackers המחוברים, ה-peers הזמינים ואת מצב ההורדה.
* העלאת קבצי .torrent ישירות ל-tracker וחסימת משתמשים במידה ונשלח קובץ שאיננו תקין.

המערכת תאפשר הפקת סטטיסטיקות בשרת ניהול (tracker) – תצוגה איכותית של מספר ההודעות והבקשות המתקבלות בכל שניה, גרפים שלפיהם יהיה ניתן להבין את ההתנהגות של האלגוריתם החוסם משתמשים "מציקים" ששולחים הודעות רבות ללא תכלית, או מעלים קבצי torrent פגומים וגם אפשרות לחסום באופן ידני משתמשים לפי הצורך דרך חלון הסטטיסטיקות.

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

* 1. אלגוריתמים עיקריים

אלגוריתמים מרכזיים במערכת

סדר האלגוריתמים שלך לא ברור! קודם יש עמית שרוצה קובץ, הוא פונה למישהו?! על מנת לבדוק איפה הקובץ שלו נמצא – זה יכול להיות בקבוצה שלו ויכול להיות מחוץ לקבוצה. מישהו עושה את זה עבורו, תוך התייעצות עם שרת המידע. לאחר מכן חוזר המידע למשתמש והוא פונה ישירות לכל שרת ניהול, אשר מחזיק TRACKER והוא פונה לכל המשתמשים בקבוצה לחבר אותם אחד אחד P2P למשתמש המבקש וכו', אנא עשה סדר באלגוריתמים ותאר אותם לפי הסדר ב BULLETS.

1. אלגוריתם למציאת הסטטוס של הקובץ

אלגוריתם זה אחראי על פענוח קובץ ה-.torrent שנטען למערכת.

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, לדוגמה במידה וקיים יותר מקובץ אחד עליו מורה, לא נתון גודל כל הקבצים ביחד, אלא רק באופן "length: {file name}", אופן זה מקשה על הקישור בין הקבצים ל-pieces, ולכן יש צורך באלגוריתם זה, שבין היתר יחבר למבנה נתונים מתאים את כל הגדלים של כל הקבצים כסדר הקבצים בקובץ ה-.torrent, כך שניתן יהיה לעבור על כל ה-pieces ולדעת אילו pieces לרשות המשתמש. שמת לב שאין אף נקודה בכל ה 10 שורות האלו?

סדר פעולות האלגוריתם:

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

האם לא היית צריך להזכיר את קיומו של שרת המידע באלגוריתם זה?

1. אלגוריתם למציאת עמיתים לצורך שיתוף חלקים, כולל לצורך השלמת חלקים מהרשת החיצונית

לאחר פענוח קובץ ה-.torrent שנטען למערכת, תפנה המערכת לשרת הניהול הפנימי ותשאל אותו האם קיימת קבוצה המשתפת את ה-torrent אצלו, במידה וכן, תברר עם כל העמיתים זה לא נמצא כבר ב DB? למה לפנות לעמיתים? מה הם החלקים שברשותם, תשמור אותם במחלקת המשתמש ותתקדם לשלב ההתחברות אליהם אך במידה ולא, או יסתבר 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 לסוגו (HTTP, UDP) ותשלח בקשה למי? להשגת משתמשים משתפי הקובץ (משתמשי הקבוצה אצל ה-tracker), תעשה זאת עבור כל tracker בקובץ
* תקבל את רשימת העמיתים מכל ה-trackers, ותשמור אותם במחלקת המשתמש, תמשיך לשלב ההתחברות אליהם

1. מציאת העמית הרלוונטי להרכבת קובץ מטרה, וקישור בין עמיתים והחלקים שברשותם

האלגוריתם אחראי על הקישור בין העמיתים לבין החלקים שברשותם. לאחר טעינת קובץ ה-.torrent ולאחר מציאת העמיתים, האלגוריתם ישלח הודעות "handshake" לכל העמיתים שהוספו למבנה נתונים במחלקת המשתמש בשלב הקודם, שהם כל העמיתים מכל הקבוצות שהובאו לו, הן מהרשת הפנימית, והן מהרשת החיצונית (במידה ובוצעו בקשות ל-trackers ברשת החיצונית), יעשה זאת במקביליות, לצורך מיפוי בלבד ולא לצורך השגת pieces מהם, ויעדכן את רשימת החלקים הזמינים לפי הודעת ה-bitfield (חלקים זמינים) שלאחר הודעת ה-handshake, handshake ו-bitfield אלה אינם קשורים ל-handshake וה-bitfield שבאלגוריתם השגת החלקים שכן תתבצע התנתקות והתחברות מחדש לאותו עמית לאחר מכן כדי להשיג piece, בהתחברות הבאה לעמית זה, לא ייעשה ב-bitfield שימוש (עדכוני ה-have מהעמית יורו על קיומו של piece חדש אצלו). טוב שיש נקודה אחת בסוף, אולי אפשר גם באמצע? אחת? שתיים? לאחר מכן בעזרת המידע שיושג מאלגוריתם זה ניתן יהיה לעבור מהחלק הנדיר לנפוץ, זאת משום שחלקים נדירים בעלי סיכוי גבוה יותר להיעלם בקרוב.

1. אופן התקשרות עם עמיתים

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, יפתח תקשורת עם עמית רנדומלי מרשימת העמיתים שמחזיקים את ה-piece, ב-thread, ימשיך לעשות זאת עד שיהיו 5 תקשורת מקביליות עם עמיתים שונים שאחראים על השגת pieces שונים
* יחכה בלולאה פנימית למקום פנוי נוסף, ימשיך בפעולות אלה עד לסיום כל תהליכי התקשורת הפתוחים.

1. תקשורת בין עמיתים לצורך השגת חלקים

באמצעות האלגוריתם מתבצעת התקשורת המורה על רצונות משתמש המערכת 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 שישיג 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

במידה ומשתמש יוצא מהמערכת, אין צורך להתנתק מהקבוצה באופן מיוחד, שרת הניהול ימחק את המשתמש בצורה אוטומטית לאחר שיזהה כי הפסיק לשתף pieces עם עמיתים אחרים.

1. בניית קבצים לפי החלקים שאוחדו

האלגוריתם יאפשר בנייה של קבצים לפי ה-pieces שהורדו עד כה במידה ומרכיבים קובץ שלם אחד לפחות, כאשר יפעל ברצף, זאת אומרת שכאשר קיימים לדוגמה החלקים A, B, D, הוא לא יתחיל בבניית הקובץ עד שחלק C יורד ויאוחד במלואו, ולאחר מכן יבנה את הקובץ במידה ואכן חלקים A, B, C, D מרכיבים **לפחות קובץ אחד**, אם לא, ימשיך ברצף עד שיהיה בבעלותו מספיק חלקים מאוחדים בשביל קובץ, הוא ימשיך ברצף זה עד שיורדו כל הקבצים הדרושים, זאת משום שהחלקים מסודרים בצורה של רצפי קבצים לפי הפרוטוקול BitTorrent, כסדר החלקים, כך גם סדר הקבצים.

1. התגברות על תקלות

זהו למעשה יישום של מספר אלגוריתמים, תפקיד כל האלגוריתמים הללו 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 לשם 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
* יבדוק 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.

במידה ולאחר 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.

1. זיהוי התנהגות לא נאותה ברשת

אלגוריתם זה יבוא לידי ביטוי בשרת הניהול וב admin, תפקידו הוא למנוע ממשתמשים לנסות ולפגוע ביציבות המערכת. מדוע ב ADMIN?

**האלגוריתם בשרת הניהול**: כאשר משתמש "מציק" לשרת הניהול, כלומר שולח לו הודעות חוזרות ונשנות שלא כצורך, יזהה האלגוריתם בעזרת סטטיסטיקות שרת הניהול את הנעשה, ויחסום באופן אוטומטי את המשתמש מכל התקשרות נוספת עם שרת הניהול.

תחום נוסף אצל שרת הניהול בו האלגוריתם יכול לבוא לידי ביטוי הוא תחום העלאת קבצי ה-.torrent אליו בידי משתמשים. כחלק מהפרויקט תתאפשר גם העלאת קבצי .torrent לשרת הניהול לצורך יצירת קבוצה חדשה. 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, חשוב לציין שתהיה גם הגבלה על מספר הקבוצות עליהם אחראי שרת הניהול, כדי לשמור על יציבות שרת זה.

**האלגוריתם ב admin:** ה admin עוקב אחר הנעשה בקבוצה, כאשר בידיו גישה למסד נתונים משותף בינו לבין שרת הניהול. באמצעות נתונים ממסד נתונים זה, יודע ה admin על מצבם של כל העמיתים בקבוצה, במידה ויזהה כי עמית איננו משתף pieces שבבעלותו עם עמיתים אחרים בקבוצה, כלומר "משקר" לעמיתים השונים על ה-pieces שבבעלותו, ייחסם המשתמש מהקבוצה באופן אוטומטי בידי ה-admin, אם זה אוטומטי, מי מפעיל את הפונקציה הזו? ולא יוכל לתקשר יותר עם שרת הניהול האחראי על הקבוצה ממנה נחסם.

* 1. דרישות ומגבלות מערכת

למערכת מספר דרישות ומגבלות.

בהיבט הדרישות: ריווח BULLETS לא נכון

* המערכת חייבת להיות מהירה, וחייב להיות ניצול מלא של רוחב הפס, הורדה מקבילית כל הזמן מכל העמיתים מהם אפשר להוריד, אך חייבת להיות הגבלה כמה? על מספר העמיתים מהם מורידים במקביליות שכן הורדה מיותר מדי עמיתים תשתמש ביותר מדי משאבים.
* מסד הנתונים המשותף בין המנהל לשרת הניהול חייב להיות מעודכן תמיד, למה שלא יהיה? אני מבין שה admin רק קורא אותו... זאת על מנת שיהיה ניתן לקבל תמונה רחבה ועדכנית של המתרחש בקבוצה, וגם לוודא ניצול אופטימלי של האלגוריתם לחסימת משתמשים מקבוצה על בסיס אופן העזרה שלהם למשתמשים אחרים אצל המנהל. מה התרחיש שבמסגרתו הם לא מסונכרנים?
* המערכת חייבת לתמוך בהורדת חלקים מן הרשת החיצונית על מנת לאפשר הורדה מלאה של קובץ ובאופן רציף.
* תמיכה בכל סוגי קבצי התוכן
* טיפול משמעתי 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 איישם, 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 איישם כחלק מהרשת הפנימית יהיה UDP.
* רק admin אחד יוכל להתחבר בכל עת לכל שרת ניהול, בשל כמות האפשרויות הקיימות

ב-admin, והאפשרות שיווצרו התנגשויות

* תמיכה במערכות הפעלה מסוג Windows 7, Windows 10 בלבד
  1. 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.

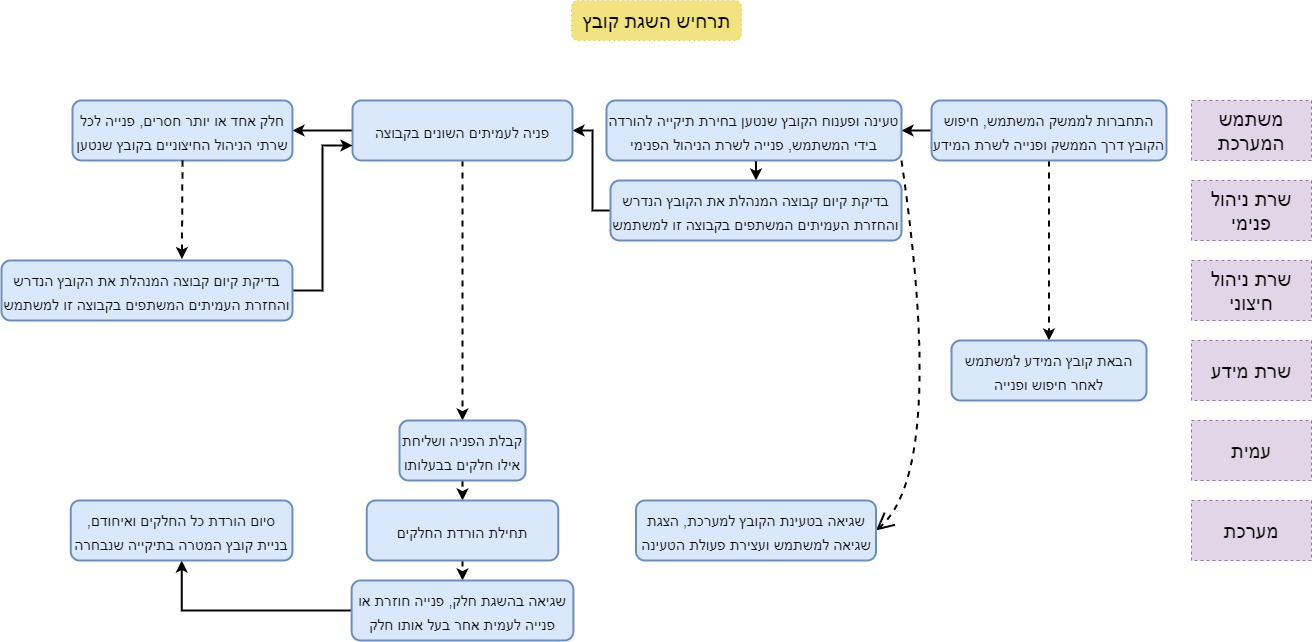
הגדרות הפרוטוקול בהם ייעשה שימוש:

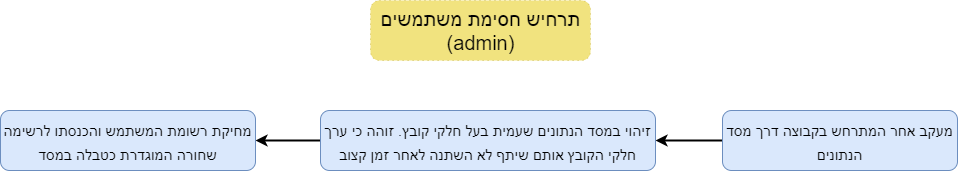
* Trackers
* Peers
* קובץ .torrent
* Peer wire protocol – אופן התקשורת בין עמיתים (אילו הודעות לשלוח, למי, מתי)
* Bencoding
* Rarest First piece downloading strategy – הורדת pieces מה-piece הנדיר, שבבעלות הכי פחות עמיתים, ל-piece הנפוץ, שבבעלות הכי הרבה עמיתים

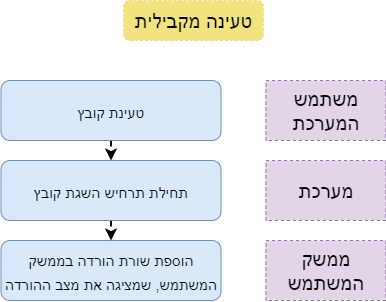
ייעשה גם שימוש ב-API של1337x לצורך חיפוש torrents כחלק מתכונת החיפוש בממשק המשתמש, במידה ולא נמצא קובץ ה-.torrent ברשת הפנימית (משרת המידע הפנימי).

* 1. 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
  1. ממשק משתמש
  2. תרחישים עיקריים





יפה! יש להוסיף טיפול בשגיאות.

1. תהליך כתיבת הפרויקט
   1. תהליך הפרויקט
   2. אתגרים ואופציות שונות למימוש
2. מרכיבי פתרון
   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 בספרייה Tkinter לבניית ה-GUI של ה-client, וגם ב-matplotlib לתצוגה גרפית של מנהל
* אבטחה – שם משתמש וסיסמה אצל המנהל, מסד הנתונים המשותף בינו לבין שרת הניהול בעל קוד אבטחה
* מבנה נתונים – שימוש גדול במערכים וב-Dictionaries כחלק מפענוח קובץ ה-.torrent, שמירת מידע על עמיתים והחלקים שברשותם אצל המשתמש, מעקב אחר ה-pieces שהורדואצל 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 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 קישור בין peers ל-pieces שברשותם באמצעות threads ויישום שונה בצורה אסינכרונית באמצעות הספרייה asyncio.
  1. סביבת העבודה (טכנולוגיה)
* שפות התכנות:

המערכת ברובה תיכתב 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.

אך יעשה שימוש גם ב-SQL למסד הנתונים של המנהל, וכן שימוש ב-C++ כדי ליישם אלגוריתם למציאת הסטטוס של הקובץ,אשר ב-Python עלול לקחת ימים להרצה.

* סביבות פיתוח:

PyCharm – לשפת python.

Visual Studio Code – לשפת C++.

* 1. מבט טופולוגי
  2. מבנה נתונים
* המבנים העיקריים ותוכנם
  1. מסד נתונים
* סכמה כללית של הישויות והקשרים
  1. מבט מודולרי
  2. פירוט מודלים עיקריים

1. תסריטי בדיקה
   1. דגשים בבדיקה
   2. תסריטי בדיקה עיקריים
2. רפלקציה
   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 שרת הניהול, המשך עבודה על ממשק המשתמש וטיפול בבאגים. |
| מאי- | סיום כתיבת ממשק המשתמש, סיום כתיבת ממשק שרת הניהול.  בדיקה סופית של הפרויקט עם כל האלמנטים שבו. |

יש לכתוב את הטבלה מחדש ללא סימני שגיאה אדומים של WORD.

* 1. אתגרים ותרומה אישית
  2. תובנות

1. הוראות התקנה ותפעול
   1. תצורה ודרישות קדם
   2. התקנה
2. ביבליוגרפיה

במהלך כתיבת הפרויקט הסתמכתי על מספר מקורות מידע:

<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/71/how-to-write-a-bittorrent-client-part-2/>

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

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

1. נספחים

הצעת פרויקט:

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