**שרת ולקוח לשיתוף קבצים חברתי**

**מבוא:**

עד שנת 1999 והמצאת נפסטר, כל לקוח שרצה לשתף קובץ ברשת למספר רב של אנשים היה צריך להעלות אותו לשרת מרכזי. אנשים שרצו להוריד את הקובץ היו ניגשים לשרת והוא היה שולח אליהם אותו. כתוצאה מכך על השרת נוצר עומס רב של בקשות ותגובות ולכן השרת היה נהיה יקר או איטי ולכן קצב ההורדה לא היה מהיר. בנוסף, היו נשמרים בו הרבה קבצים שהצטברו למידע רב וגרמו לשרת להיות יקר.

לכן, בשביל לנצל את היכולת של השיתוף החברתי נוצרו תוכנות כגון נפסטר וביטורנט. הלקוחות שמשתמשים בביטורנט לדוגמא, נעזרים בשרת מרכזי שיקשר ביניהם. השרת מודיע למורידים ממי להוריד ולמעלים למי להעלות. כך המידע של הקבצים היה נשמר אצל הלקוחות והשרת לא היה צריך לשלוח מידע רב להרבה אנשים. בכך יצרנו מערכת זולה יותר והרבה יותר יעילה.

אני הולך לבנות שרת ולקוח המתבססים על עקרונות הביטורנט בשפת Python ובסביבת העבודה של PyCharm. אני אבנה את הפרוטוקולים בדרך שלי ואבנה את כל המרכיבים המרכזיים שהשרת והלקוח צרכים על מנת שהמערכת תעבוד.

**הסבר כללי:**

התוכנה עובדת עם קבצים בעלי הסיומת yftf. קבצים אלו הם למעשה מצביעים שמטרתם להפנות את התוכנה למידע הדרוש על מנת להוריד את הקובץ המבוקש. קובץ ה-yftf מפנה את התוכנה ל-Tracker, שהוא למעשה שרת המכוון את כל התנועה בין המשתמשים המורידים את אותו קובץ.

התוכנה שולחת ומקבלת מספר חלקים מקובץ ההורדה בו-זמנית, על פי הוראות ה-Tracker. היכולת לשלוח ולקבל חלקים שונים של הקובץ בו זמנית, ממשתמשים שונים, מאפשרת לתוכנה לשמור על קצב העברה גבוה מאוד.

**דוגמא להורדת קובץ שכבר נמצא ברשת:**

* אדם מסוים רוצה להוריד את הקובץ A באמצעות התוכנה.
* הוא תר את האינטרנט אחר קובץ yftf המצביע על הקובץ A, ומפנה את התוכנה לשרת Tracker.
* ברגע שתוכנה מתחברת לשרת, היא מקבלת ממנו הוראות באשר למי מהמשתמשים השונים עליה להתחבר.
* התוכנה מתחברת למשתמשים רבים ומורידה מכולם את הקובץ בו-זמנית.
* כל חלק מהקובץ שהורד והוא כבר מוכן, הופך להיות זמין למשתמשים אחרים דרך התוכנה. כלומר, ברגע שהתוכנה מסיימת להוריד חלק מסוים, היא חולקת אותו עם שאר המשתמשים על ידי ידוע שרת ה-Tracker.
* בו בזמן, השרת מפנה משתמשים אחרים אל המחשב של אותו אדם, על מנת שיוכלו להוריד ממנו חלקים מהקובץ שהורדתם הסתיימה.

**דוגמא להעלאת קובץ חדש לרשת:**

* אדם מסוים רוצה להעלות את הקובץ A באמצעות התוכנה.
* התוכנה יוצרת בשבילו קובץ yftf שמכיל את הפרטים עליו וכתובת שרת ה-Tracker.
* בשרת ה-Tracker הקובץ A נרשם והשרת מתחיל לנהל את האנשים שמורידים ומעלים אותו.
* אותו אדם מפרסם את קובץ ה-yftf ברשת.
* לקוחות חדשים שמורידים את הקובץ A מצטרפים לרשימה הלקוחות בשרת ה-Tracker.
* אותו אדם ששיתף את הקובץ A, מעלה אותו למורידים החדשים עד שהוא מעלה את כל הקובץ למספר מורידים חדשים שיוכלו להחליף אותו בעבודתו להעלות את הקובץ למורידים חדשים.
* כאשר הם מסיימים להוריד את הקובץ A (או חלקים ממנו) הם מתחילים להעלות אותו לאנשים חדשים שמורידים אותו.

**מרכיבים:**

* **מעלה ראשוני-** האיש הראשוני שמעלה את הקובץ החדש לרשת. הוא מעלה את הקובץ למורידים הראשונים.
* **קובץ yftf-** מכיל מידע על הקובץ אותו משתפים ועל כתובות שרתי ה-Trackers שעוקבים אחרי שיתוף הקובץ ברשת.
* **שרתים ברשת-** הם אלו שמכילים את קבצי ה-yftf ולקוחות שרוצים להוריד קובץ מסוים, מורידים מהם את קובץ ה-yftf שמפנה אותם לשרתי ה-Tracker שעוזרים להם להוריד את הקובץ במהירות בזכות השיתוף בין האנשים.
* **שרתי Tracker-** שרתים שמקשרים בין המעלים והמורידים של קבצים מסוימים שהם מטפלים בהם. הם גורמים לשיתוף של הקובץ בין אנשים רבים ובכך הם גורמים להורדה מהירה יותר של הקבצים. הם גורמים לאנשים שכבר הורידו קובץ מסוים (חלקו או כולו), להעלות חלקים ממנו לאנשים אחרים שמורידים את אותם חלקים.
* **לקוחות שמורידים קבצים-** נמצאים ברשימות הלקוחות שבשרתי ה-Tracker כמורידים של הקובץ אותו הם מורידים. שרתי ה-Tracker מקשרים אותם לאנשים שמעלים את אותו הקובץ (חלקים ממנו שלבסוף יהיו כל הקובץ). יכולים גם בו זמנית להעלות חלקים מהקובץ שהם כבר הורידו לרשת לאנשים אחרים.
* **לקוחות שמעלים קבצים-** נמצאים ברשימות הלקוחות שבשרתי ה-Tracker כמעלים של הקובץ אותו הם מעלים והם מעלים חלקים ממנו לאנשים שמורידים את הקובץ ושרתי ה-Tracker מפנים אותם אליהם. יכולים גם בו זמנית להוריד את אותו הקובץ מהרשת ומאנשים אחרים.

**פירוט תוכנה:**

**לקוח:**

* **משתף קובץ חדש-** תהיה אפשרות לשתף לרשת קובץ חדש. הלקוח יבחר באפשרות זאת ויבחר את הקובץ שהוא רוצה להעלות שנמצא במחשב שלו. התוכנה תייצר לו קובץ yftf שיכיל את המידע על הקובץ החדש והוא יכיל את הכתובת לשרת ה-Tracker של התוכנה בלבד (משום שלתוכנה יהיה פרוטוקול משלה שלא יוכל לתקשר עם שרתי Tracker אחרים ברשת). התוכנה תקשר עם שרת ה-Tracker ושרת ה-Tracker יוסיף רשימה חדשה של המעלים והמורידים של הקובץ החדש. הלקוח ישתף את קובץ ה-yftf באינטרנט ויהיה עליו להעלות את הקובץ החדש למורידים החדשים עד שמספר אנשים יורידו את כולו (אם הוא לא יעלה אותו לאנשים לא יהיה שיתוף של הקובץ בין כולם ולכן שיתוף הקובץ ברשת לא יעבוד).
* **מוריד-** לקוח מוריד מהאינטרנט קובץ yftf ותהיה בתוכנה אפשרות להורדת קובץ. כאשר הוא יבחר באפשרות זאת ויבחר בקובץ ה-yftf, התוכנה תצרף אותו לשרת ה-Tracker לרשימה של הלקוחות של אותו קובץ כמוריד. ואז התוכנה תמשיך לתקשר עם שרת ה-Tracker והוא יפנה אותה לחיבור עם אנשים שמעלים חלקים מאותו קובץ והלקוח יוריד מהם את החלקים שהם משתפים עד שלבסוף יבנה הקובץ כולו. התוכנה תבדוק את האמינות החלק שהיא הורידה על ידי הנתונים שנמצאים בקובץ ה-yftf, אם הוא לא תקין היא תפיל אותו ותתחיל להוריד אותו מחדש מלקוח אחר שמעלה את אותו החלק של הקובץ. בו בזמן הלקוח יוכל להעלות חלקים מהקובץ שהוא כבר הוריד.
* **מעלה-** ברגע שלקוח הוריד את החלק הראשון מהקובץ, התוכנה מתקשרת עם שרת ה-Tracker ובו הלקוח יתווסף לרשימת הלקוחות כמעלה הקובץ. כל עוד הלקוח מוריד את הקובץ ולא עוצר את העלאתו כאשר הסתיימה ההורדה, הלקוח יעלה חלקים מהקובץ שהוא כבר הוריד לאנשים שמורידים את אותם חלקים. שרת ה-Tracker יגיד ללקוח איזה חלק מהקובץ להעלות ולאיזה כתובת. ברגע שהוא מסיים להעלות לאותה כתובת הוא חוזר על אותו התהליך כל עוד הלקוח נותן לתוכנה להמשיך להעלות.

**שרת ה-Tracker:**

הוא אחרי על קישור המורידים והמעלים של הקבצים ברשת בהם הוא מטפל. הוא מתקשר עם התוכנות של הלקוחות ומודיע להם לאיזה כתובות להעלות איזה חלקים של קובץ מסוים ואיזה חלקים של אותו הקובץ להוריד מאיזו כתובת. בשיתוף קובץ חדש השרת יבנה רשימה חדשה של מורידים ומעלים של הקובץ. כאשר יתווספו מורידים של הקובץ, תוכנת הלקוח תודיע לשרת ה-Tracker והוא יוסיף אותם לרשימת הלקוחות כמורידים שבה ירשם איזה חלקים הלקוח כבר הוריד. כאשר יתווספו מעלים של הקובץ, תוכנת הלקוח תודיע לשרת ה-Tracker והוא יוסיף אותם לרשימת הלקוחות כמעלים. בנוסף, במשך כל תהליך ההורדה, השרת עוקב על איזה חלקים הלקוח כבר הוריד והוא שומר אותם ברשימה. מכאן השרת יצור ביעילות המרבית את שיתוף הקובץ (חלקיו) ברשת בין המורידים למעלים.

**קבצי yftf:**

מורכב מהנתונים הבאים (המחרוזות מקודדות ב-UTF-8):

* **Announce-** כתובת ה-URL של שרת ה-Tracker.
* **Info-** ספרייה שמכילה את כל הנתונים הבאים על הקובץ או הקבצים המשותפים:
  + **Name-** שם הקובץ או התיקייה (שמכילה את הקבצים) המשותפים.
  + **Piece Length-** אורך כל חלק של כל קובץ או קבצים ב-Bytes (חוץ מהחלק האחרון שיכול להכיל פחות Bytes). במקרה ויהיה מספר קבצים שונים, החלקים יוכלו להכיל קבצים שונים. אורך כל חלק יהיה 256 KiB.
  + **Pieces Hash-** (במקרה של שיתוף קובץ אחד בלבד) רשימה של Hash-ים של כל חלק בקובץ. ה-Hash יהיה מסוג SHA-1. כל Hash יהיה באורך 160 Bit.
  + **Length-** (במקרה של שיתוף קובץ אחד בלבד) אורך הקובץ המשותף ב-Bytes.
  + **Hash-** (במקרה של שיתוף קובץ אחד בלבד) Hash מסוג SHA-1 באורך 160 Bit שיתבצע על הקובץ.
  + **Files-** (במקרה של שיתוף יותר מקובץ אחד) רשימה שמכילה ספריות לכל קובץ שמכילות את הנתונים הבאים על כל קובץ:
    - **Path-** מחרוזת של מיקום הקובץ בתוך התיקייה הראשית.
    - **Length-** אורך הקובץ המשותף ב-Bytes.
    - **Hash-** Hash מסוג SHA-1 באורך 160 Bit שיתבצע על כל קובץ.
    - **Pieces Hash-** רשימה של Hash-ים של כל חלק בקובץ. ה-Hash יהיה מסוג SHA-1. כל Hash יהיה באורך 160 Bit.

**פרוטוקולי YFT:**

הלקוח ושרת ה-Tracker יוכלו לנהל מספר שיחות במקביל בעזרת שימוש ב-select (ספרייה ב-Python העוזרת לנהל מספר שיחות במקביל) וב-Tornado (ספרייה ב-Python לשרת ולקוח שפועלים על פרוטוקול HTTP שעל TCP בצורה אסינכרונית).

**בין הלקוח לשרת ה-Tracker:**

חלק ראשון בשרת ה-Tracker. החלק שמנהל את השיחה בין הלקוחות שמעלים ומורידים לבין השרת.

בבקשה הלקוח שולח פרטים על עצמו, על הקובץ שהוא רוצה להוריד ועל מצב ההורדה/העלאה של הלקוח בנוגע לקובץ המסוים. אם הוא מוריד הוא מודיע על הפורט שעליו הוא מאזין ומחכה למידע והוא גם מודיע על אינדקס של חלק שהוא רוצה להוריד. על הדרך הוא גם מודיע על אינדקס של חלק שהוא סיים להוריד (אם יש). בנוסף, הלקוח גם יכול להודיע אם הוא יכול להעלות.

בתגובה או שהשרת יחזיר הודעת שגיאה אם הייתה, או שיחזיר את המידע שהלקוח ביקש. המידע יכיל תמיד את מזהה הקובץ, סוג התגובה (אם למעלה או מוריד) וכתובת ip. אם סוג התגובה היא למוריד אז כתובת ה-ip תפנה אותו ללקוח שיעלה אליו את המידע המבוקש על הפורט שנשלח בבקשה. אם סוג התגובה היא למעלה אז כתובת ה-ip תפנה ללקוח המוריד והפורט יפנה לפורט שעליו מאזין הלקוח האחר. בנוסף יצוין החלק שאותו הלקוח יצטרך להעלות ללקוח המאזין בפורט שצוין. מציאת המעלה המתאים תתבצע מהמידע שנמצא באובייקטים השמורים בשרת.

פרוטוקול התקשורת מבוסס על פרוטוקול HTTP שעל TCP.

**כותרות הבקשה:**

YFT-Upload-Piece ו-YFT-Request-Piece-Index לא יכולים להיות באותה בקשה יחד. אם יהיו, השרת לא יתייחס ל-YFT-Upload-Piece.

* **YFT-Info-Hash-** Hash מסוג SHA-1 באורך 160 Bit שיתבצע על חלק ה-Info בקובץ ה-yftf. זה הוא המזהה של הקובץ שהלקוח רוצה להוריד או להעלות.
* **YFT-Peer-id-** מחרוזת רנדומלית באורך 20 תווים המכילה את מזהה הלקוח.
* **YFT-Peer-Status-** מצב ההורדה/העלאה של הלקוח בנוגע לקובץ המסוים. אם 0, זאת אומרת שהלקוח מתחיל להוריד את הקובץ ולכן השרת יצרף אותו לרשימת המורידים/מעלים (השרת יבדוק קודם אם הוא כבר לא נמצא שם). אם 1 זאת אומרת שהלקוח כבר מוריד או מעלה והוא נמצא ברשימה. אם 2, השרת ימחק את הלקוח מרשימת המורידים/מעלים והוא יתעלם משאר הכותרות.
* **YFT-Port-** (במקרה והלקוח עדיין מוריד) מספר הפורט שעליו הלקוח מאזין ומחכה למידע מלקוח אחר שמעלה.
* **YFT-Request-Piece-Index-** (במקרה והלקוח עדיין מוריד) האינדקס של החלק הבא שהלקוח רוצה להוריד.
* **YFT-Finished-Piece-Index-** (במקרה והלקוח עדיין מוריד וסיים להוריד חלק) האינדקס של החלק שהלקוח סיים להוריד ויכול להעלות עכשיו.
* **YFT-Upload-Piece-** (במקרה והלקוח סיים להעלות חלק/ים או יכול להתחיל להעלות חלק לראשונה) מכיל 1 אם הלקוח יכול להעלות חלק/ים.

**כותרות התגובה:**

* **YFT-Info-Hash-** Hash מסוג SHA-1 באורך 160 Bit שיתבצע על חלק ה-Info בקובץ ה-yftf. זה הוא המזהה של הקובץ שהלקוח רוצה להוריד או להעלות.
* **YFT-Error-** (במקרה ויש שגיאה) מידע על השגיאה (לאחריו לא יופיעו הכותרות הבאות).
* **YFT-Type-** יכיל 1 אם הלקוח (שאליו נשלחת התגובה) הוא מעלה ו-0 אם הוא מוריד.
* **YFT-Ip-** כתובת ה-ip של הלקוח האחר שמעלה או מוריד.
* **YFT-Piece-Index-** (במקרה והלקוח מעלה) האינדקס של החלק שאותו יעלה הלקוח ללקוח האחר.
* **YFT-Port-** (במקרה והלקוח מעלה) מספר הפורט שעליו מאזין לקוח אחר שאליו יעלה הלקוח את המידע.

**בין הלקוח המשתף קובץ חדש לשרת ה-Tracker:**

החלק השני בשרת ה-Tracker. החלק שאחראי על יצירת האובייקטים שמנהלים את המעלים והמורידים של הקובץ החדש.

הלקוח שולח בקשה לשרת שיחל לנהל את הקובץ שלו. הבקשה מכילה (באזור ה-body) את קובץ ה-yftf (שנשמר בשרת) ומידע על הלקוח (בכותרות). בתגובה השרת יבנה את האובייקט (הוא גם יחשב את מספר החלקים בקובץ) ויוסיף את הלקוח לרשימת הלקוחות באובייקט כלקוח שמעלה (שם יהיה רשום שיש בידיו את כל החלקים) וישלח לו בתגובה הודעת אישור או שגיאה אם קרתה. האובייקטים יישמרו בשרת כקבציי טקסט בעזרת שימוש בפונקציית pickle.

פרוטוקול התקשורת מבוסס על פרוטוקול HTTP שעל TCP.

**הכותרות וה-body בבקשה:**

* **YFT-Peer-id-** מחרוזת באורך 20 תווים רנדומליים המכילה את מזהה הלקוח.
* **YFT-Peer-Status-** מצב ההורדה/העלאה של הלקוח בנוגע לקובץ המסויים. יכול להכיל 0 או 1 או 2 כמו שהוסבר כותרת הבקשה של הפרוטוקול בין הלקוח לשרת ה- Tracker (במקרה זה יהיה חייב להיות 0).
* **YFT-Upload-Piece-** מכיל 1 אם הלקוח יכול להעלות חלק/ים (במקרה הזה יהיה חייב להיות 1).
* **Body-** קובץ ה-yftf שמכיל את המידע על הקובץ המשותף.

**כותרות התגובה:**

* **YFT-Info-Hash-** Hash מסוג SHA-1 באורך 160 Bit שיתבצע על חלק ה-Info בקובץ ה-yftf. זה הוא המזהה של הקובץ המשותף.
* **YFT-Error-** (במקרה ויש שגיאה) מידע על השגיאה.

**המבנה של אובייקט:**

כל הניהול של הקובץ המשותף יתבצע בעזרת שימוש באובייקט שלו שבשרת. באובייקט ישמרו מזהה הקובץ המשותף, מספר החלקים שבו והלקוחות (המזהה שלהם) שמורידים ומעלים אותו והמידע עליהם. כל עוד הלקוח לא שלח בכותרת YFT-Peer-Status 2 אז הוא יופיע ברשימה. כאשר לקוח מבקש להוריד חלק מסוים, השרת יחפש את המעלה המתאים ביותר על ידי כך שהוא יעבור על רשימת הלקוחות, יבדוק אם הם יכולים להעלות ואם הם יכולים להעלות את החלק המתאים. לקוח שיכול להעלות הוא לקוח ששלח לשרת בקשה עם YFT-Upload-Piece שבו יש 1 ולכן הסטאטוס של הלקוח יהיה 1. כאשר השרת ימצא התאמה הוא יהפוך את הסטאטוס של הלקוח ל-0 עד שהלקוח ישלח את הבקשה מחדש. בנוסף השרת ישמור רשימה של כל החלקים שהלקוח הוריד וישמור את כתובת ה-ip שלו.

**דוגמא לשני אובייקטים:**

|  |  |
| --- | --- |
| Info-Hash | Info-Hash |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | [Pieces that he can upload] | [Peer-id, Peer-ip, State] | | [0, 1, 2] | [Peer-id, 10.1.1.3, 1] | | [0, 1, 3] | [Peer-id, 10.1.1.4, 0] | | Num-Pieces | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | [Pieces that he can upload] | [Peer-id, Peer-ip, State] | | [0, 2] | [Peer-id, 10.1.1.1, 0] | | [0, 1, 2, 3] | [Peer-id, 10.1.1.2, 1] | | Num-Pieces | |

**בין הלקוח המוריד ללקוח המעלה:**

תקשורת בין שני לקוחות ששרת ה-Tracker קישר ביניהם, אחד המוריד והאחר המעלה. שניהם מקבלים מהשרת את כתובת ה-ip אחד של השני ובנוסף המעלה מקבל את הפורט שעליו המוריד מאזין ואת האינדקס של החלק שהוא צריך להעלות. המעלה מתקשר עם המוריד והם יוצרים קישור ומתחילים שיחה. המעלה שולח למוריד את ה-Info-Hash ואת המידע של החלק. כאשר הכול נשלח בהצלחה השיחה מסתיימת.

השיחה היא על פרוטוקול P2P שעל TCP.

**מבנה המידע:**

* **Info-Hash-** Hash מסוג SHA-1 באורך 160 Bit (כ-40 תווים כי בבסיס 16) שיתבצע על חלק ה-Info בקובץ ה-yftf. זה המזהה של הקובץ המשותף שהמעלה שולח למוריד.
* **Data-** המידע שמכיל החלק שאותו צריך המעלה לשלוח למוריד (המוריד יודע את אורך המידע משום שזה מופיע בקובץ ה-yftf).

**דומה ושונה מ-BitTorrent:**

**דומה:**

* הלקוחות שולחים את המידע של הקובץ המשותף אחד לשני ולא דרך שרת מרכזי.
* יש את שרת ה-Tracker שמנהל את הקישורים בין המורידים למעלים.
* יש בדיקה אם הקובץ שנשלח תקין והמידע שלו לא שונה.

**שונה:**

* הלקוח יכול להחליט את דרך ההורדה שלו. הוא יכול להחליט אם הוא מוריד כל הזמן חלקים רנדומליים או שהוא מוריד את החלקים בסדר שלהם (Streaming, שזה מה שיהיה בלקוח שלנו).
* קובץ ה-yftf מחלק כל קובץ בנפרד. אם יש כמה קבצים משותפים בו זמנית בשיתוף אחד אז הוא לא מחלק את כל המידע ביחד לחלקים אלה כל מידע של כל קובץ בנפרד. זאת על מנת שהלקוח לא יצטרך להוריד את כל הקבצים המשותפים.
* שרת ה-Tracker אומר למעלה איזה חלק לשתף ולא נותן למורידים להגיד למעלה איזה חלק לשתף. זאת על מנת לצמצם ככל האפשר את השגיאות שיכולות להיווצר בין הלקוחות.
* בעת הורדה או העלאה של קובץ מסוים, הלקוח יכול לתקשר רק עם שרת Tracker אחד של אותו הקובץ ולא עם כמה במקביל.
* חסרים כמה חלקים של השיחה בין הלקוחות שקיימים ב-BitTorrent, כגון Choke ו-Interested.
* חסרים כמה אלגוריתמים נוספים לשיתוף קבצים מהיר יותר שקיימים ב-BitTorrent, כגון DHT ו-Fast Peers Extensions.

**קוד:**

**מחלקות:**

**תיקיות וקבצים:**

* דוגמא לתיקייה
  + - דוגמא לקובץ בתיקייה

**צעדים להמשך:**

ישנם הרבה שיפורים נוספים שאפשר להוסיף לפרויקט שלי ויכול להיות שאוסיף אותם בהמשך:

* ליישם את השרת והלקוח והפרוטוקולים שלהם.
* ליישם אלגוריתמים נוספים לשיתוף יעיל של הקבצים המשותפים.
* ליישם דרכים נוספות לשיתוף קבצים מהיר יותר, כמו DHT.
* ליישם פרוטוקול P2P שעובד גם מאחורי שרתי NAT.
* ליישם שיחה יעילה יותר בפרוטוקול P2P כך שלא ייוצרו תקלות ותקיעות מיותרות.
* לאבטח יותר את השיחה בין הלקוחות כך שלא יהיו בעיות אבטחה.
* ליישם לקוח שיוכל לתקשר עם כמה שרתי Tracker במקביל בנוגע לקובץ מסוים שהוא רוצה להוריד או להעלות.
* ליישם לקוח שיוכל להוריד או להעלות כמה קבצים במקביל.
* ליישם לקוח שאם הוא יוצא מהתוכנה וחוזר אליה אז הוא יוכל להמשיך להעלות קבצים ישנים שהוריד ועדיין נמצאים על המחשב שלו.

**סיכום:**

במהלך הפרויקט חקרתי ולמדתי הרבה דברים חדשים:

* חקרתי את BitTorrent לעומק, למדתי על הפרוטוקולים שלו ועל איך שהוא עובד.
* חקרתי את פרוטוקול P2P ולמדתי על הבעיות שבו.
* למדתי הרבה דברים חדשים על שיתוף קבצים חברתי ברשת.
* למדתי איך לחקור נושאים חדשים שאני לא מכיר.
* למדתי איך לתכנן בניית פרויקטים כך שהבנייה תתבצע בצורה יעילה.

**ביבליוגרפיה:**

<http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1/>

<http://bittorrent.org/beps/bep_0003.html>

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

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

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

<http://www.howtogeek.com/howto/31846/bittorrent-for-beginners-how-get-started-downloading-torrents/>

<http://www.ccs.neu.edu/home/amislove/teaching/cs4700/fall09/lectures/lecture20.pdf>

<https://he.wikipedia.org/wiki/%D7%91%D7%99%D7%98%D7%95%D7%A8%D7%A0%D7%98>

<http://www.kaspersky.com/au/images/camilo_andr%D1%83s_gonzalez_toro.pdf>

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