Communication Networks - Final Project

מגישים: 215312729, 318790169, 209092287

קישור להקלטות pcap:

https://drive.google.com/drive/folders/1aF9--1N4EBlxPKddHXaTYQeovzDNOXpL?usp=sharing

קישור לgithub:

https://github.com/Wahylian/NetComFinalProj.git

חלק ראשון:

1. **משתמש מדווח על מהירות העברת קבצים נמוכה, ואנו צריכים לחקור את שכבת התעבורה לסיבות פוטנציאליות.**

הסיבות שיכולות להשפיע על כך הן:

א. במקרה שבו:

- הרשת שבה הלקוח משתמש אינה אמינה עם סיכוי גבוה לאובדן מידע
- הלקוח מנסה להעביר כמות גדולה של מידע

והלקוח משתמש בפרוטוקול UDP או QUIC לצורך העברה, יכולה להיות האטה משמעותית במהירות העברת הקבצים.

הסיבות לכך הן:

- UDP ו QUIC אינם פרוטוקולים אמינים עם סיכויים לאובדן מידע והגעתו בסדר הלא נכון.
- במצב שבו אין רשת אמינה או שהלקוח מנסה להעביר כמות גדולה של מידע, באופן הסתברותי כאשר משתמשים בפרוטוקולים עם סיכויים לאובדן מידע כמות המידע ש"הולך לאיבוד" בדרך וקיים צורך שישלח מחדש הינה גדולה מספיק בשביל להאט באופן משמעותי את מהירת העברת הקבצים.

ניתן לבדוק אם זה המצב כך:

- בדיקת גודל הקובץ שהלקוח מנסה להעביר.
- בדיקת סוג הרשת שעליה הלקוח מנסה להעביר את הקובץ (סיבים אופטיים, קו טלפון וכדומה).
- להסניף את סוג החבילות שהלקוח מנסה להעביר.

במידה והלקוח אכן ניסה להעביר כמות גדולה של מידה או שהסוג רשת שאליה הוא ניסה להעביר את המידע היא אכן לא אמינה והוא משתמש ב UDP או ב QUIC, נעביר אותו להשתמש בפרוטוקול TCP מכיוון ש:

- TCP הוא פרוטוקול אמין שלא מאפשר אובדן מידע ולכן הוא חוסך את זמן השידור החוזרים שקורים ב UDP ו QUIC.

ב. במקרה שבו:

- הרשת שבה הלקוח משתמש אמינה עם סיכוי נמוך לאובדן מידע
- הלקוח מנסה להעביר כמות קטנה של מידע

והלקוח משתמש בפרוטוקול TCP לצורך העברה, יכולה להיות האטה משמעותית במהירות העברת הקבצים.

הסיבות לכך הן:

- TCP הוא פרוטוקול העברת מידע אמין ששולח מידע אך ורק בסדר הנכון.
  בשביל שזה יתקיים, הוא מוודא שכל חבילה מגיעה בעזרת ACK לפני שהוא שולח חבילה נוספת, מה שמגדיל את הזמן הכולל שלוקח לכל המידע לעבור.
- בתחילת כל תקשורת חדשה בין הלקוח לשרת, הפרוטוקול מבצע לחיצת ידיים משולשת לפני שהוא יכול להתחיל להעביר מידע.

כאשר כמות המידע שרוצים להעביר היא קטנה, לחיצה זאת לוקחת פרק זמן משמעותי מזמן העברה הכולל.

ניתן לבדוק אם זה המצב כך:

- בדיקת גודל הקובץ שהלוקח מנסה להעביר.
- בדיקת סוג הרשת שעליה הלקוח מנסה להעביר את הקובץ.
- להסניף את סוג החבילות שהלקוח מנסה להעביר.

במידה והלקוח אכן ניסה להעביר כמות קטנה של מידע או שהרשת אמינה עם מעט אובדן מידע והוא משתמש ב TCP, נעביר אותו להשתמש ב UDP או QUIC מכיוון ש:

- UDP ו QUIC מהירים יותר מ TCP מכיוון שהם לא אמינים כמוהו, שכן הם ממשיכים להעביר חבילות גם במקרה ואחת מהן לא מגיעה ליעד.
- UDP לא בודק על הגעה של חבילות ו QUIC עובד על מספר זרמים בחיבור UDP אחד כך שגם אם באחד מהם חבילה לא הגיעה יש זרמים אחרים שדרכם מידע עובר.

ג. במקרה שבו:

- יש שימוש ב TCP עם sliding window קטן (לדוגמא בגודל 1).
- כמות גדולה של מידע להעביר.

תתרחש האטה במהירות העברת הקובץ.

הסיבות לכך:

- מכיוון שגודל החלון ב TCP קובע כמה חבילות ניתן להעביר מבלי לחכות ל ACK, כאשר החלון קטן (לדוגמה בגודל 1), ניתן להעביר רק חבילה אחת עד לקבלת ה ACK ורק לאחר מכן להעביר חבילה נוספת.
- כאשר יש כמות גדולה של מידע, זמן ההמתנה לכל ACK, נהיה משמעותי ומצטבר.
- כאשר ה RTT גדול ויש הרבה חבילות להעביר, אם מעבירים רק אחת בכל פעם יקח זמן רב מאוד להעביר את כולן.

ניתן לבדוק אם זה המצב כך:

- בדיקת גודל הקובץ שהלקוח מנסה להעביר.
- מעקב אחר ה RTT.
- הסנפת החבילות שהלקוח מנסה להעביר ובדיקת גודל ה sliding window שהלקוח משתמש בו. ניתן לבדוק זאת ע"י בדיקה של כמות החבילות שהלקוח מעביר מבלי לחכות עד שהוא מקבל ACK מהשרת עליהן, ולראות את ממוצע החבילות שנשלחות בין ACK ל ACK מהשרת.
  אם הממוצע הוא קטן סימן שהחלון הוא צוואר בקבוק.

במידה והלקוח אכן מנסה להעביר כמות גדולה של מידע בעזרת TCP עם sliding window קטן, נגדיל את ה sliding window שלו, כך שהוא יוכל להעביר כמות גדולה של חבילות בין ACK לACK ובכך להגדיל את כמות החבילות שעוברות ברשת בכל רגע נתון.

ד. במקרה שבו:

- יש נתב עם באפר משותף ומספר זרמי מידע שעוברים דרכו, ביניהם המסלול מהלקוח לשרת (או להיפך), לצורך הדוגמה נניח שני זרמי מידע.
- הלקוח אינו משתמש ב TCP או QUIC.
- הלקוח מעביר מידע במהירות שמתקרבת למהירות שהנתב יכול להעביר אותו.

תתרחש האטה משמעותית במהירות העברת הקובץ.

הסיבות לכך:

- ככל שמהירות העברת המידע של הלקוח מתקרבת למהירות העברה של הנתב כך העיכוב בהעברת המידע גדל באופן אקספוננציאלי.
- חלק מהבאפר משומש עבור מידע שכבר נשלח בעבר ולא הגיע ליעד ובכך הוא "מתבזבז" במקום לשלוח מידע חדש.

- חלק מהבאפר משומש עבור מידע שכבר נשלח בעבר והגיע ליעד אך הלקוח לא ידע שהוא הגיע ליעד, בעקבות הפעלת timeout מוקדם מידי (קורה כאשר הלקוח משתמש בפרוטוקול עם timeout ללא congestion control).
- כאשר הלקוח הנוסף משתמש בנתב המשותף ושולח כמות גדולה של מידע הדורשת כמות גבוהה של משאבים מהנתב.

ניתן לבדוק אם זה המצב:

- הסנפת סוג החבילות שהלקוח משתמש בהם.
- בדיקה של הנתבים בדרך מהלקוח לשרת, בדיקת העומס עליהם, וגודל הבאפר שלהם ואת מהירות העברת המידע שלהם ביחס למהירות העברת המידע של הלקוח.
- מעקב אחר ה RTT.

במידה והלקוח אכן משתמש בפרוטוקול שונה מ TCP או QUIC וקיימת בעיה ב Congestion Control, נעביר אותו להשתמש ב TCP או QUIC שכן:

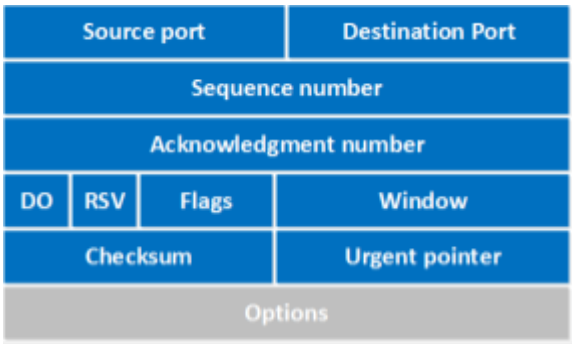
- ל TCP ול QUIC יש Congestion Control שמבוסס על AIMD ועל Slow Start בשביל להתמודד עם ה Congestion להימנע ממנו.

ובכך להאיץ את תהליך העברת המידע שכן כאשר יש Congestion העברת המידע מאטה באופן משמעותי והזמן שלוקח להעביר אותו גדל.

2. **ל TCP יש מכניזם של Flow Control, חקור את השפעתו על העברת מידע.**
איך יושפעו הביצועים שלו כאשר לספק יש כוח עיבוד גבוה משמעותית מאשר הלקוח?

נסתכל תחילה על איך המכניזם של Flow Control ב TCP עובד, ולאחר מכן נסיק מסקנות על השפעתו על ההספק כאשר לספק יש כוח עיבוד גבוה משמעותית מאשר הלקוח.
לאחר מכן נראה מה המשמעות לגבי הביצועים.

דרך העבודה של Flow Control ב TCP:



התמונה נלקחה מ
https://networklessons.com/ip-routing/tcp-header
בתמונה rwnd מיוצג ע"י הערך Window.

- ב TCP Header ישנו שדה בשם receive window (לרוב מקוצר ל rwnd).
- שדה זה מאפשר למקבל המידע לפרסם בצורה דינאמית את גודל הבאפר הפנוי שיש לו בעבור קבלת מידע ב Bytes.
- כך השולח מגביל את כמות החבילות שהוא שולח מבלי לקבל ACK לפי גודל ה rwnd שהתקבל.
- ובכך מובטח שלא ישלח יותר מידע מאשר שהבאפר יכול להתמודד איתו.
- כאשר החלון מתמלא לגמרי, הספק מפסיק לשלוח חבילות, ומתחיל לחכות ל ACK חדש מהלקוח עם גודל החלון העדכני.
- במקרה ו ACK זה לא מגיע, הספק שולח כל פרק זמן מסוים הודעה בגודל Byte אחד ומחכה לקבל ACK עליה עם גודל ה rwnd החדש.

כעת נראה מה ההשפעה של ה Flow Control על ההספק:

- מכיוון שמתקיים $W/t = P$, כאשר P זה ההספק, W זה עבודה, ו t זה הזמן שלוקח לבצע את העבודה.

ובמקרה שלנו העבודה תהיה כמות הביטים שאנו רוצים להעביר מהספק אל הלקוח, כלומר W יהיה קבוע, ולכן המשתנה היחיד שמשפיע על ההספק (P) הוא t- הזמן שיקח לספק להעביר את המידע אל הלקוח.

- נגדיר את המשתנים הבאים:
  - $U_S(Bytes/sec)$ - קצב העלאת הנתונים של הספק.

- $D_R(Bytes/sec)$- קצב הורדת הנתונים של הלקוח.
- $t(sec)$- כמות הזמן שלוקח לספק להעביר את המידע ללקוח.

- ידוע לנו שכוח העיבוד של הספק הינו גבוה משמעותית משל הלקוח, ולכן קצב העברת הנתונים שלו (במקרה הזה העלאת נתונים) יהיה גדול משמעותית משל הלקוח (במקרה זה הורדת נתונים).
ולכן מתקיים $U_S > D_R$.
- כאשר יש את ה Flow Control של TCP, הספק לא ישלח יותר חבילות גדולות יותר מאשר המקום הפנוי בבאפר, ולכן באופן מעשי קצב העלאה של הספק יהיה שווה לקצב ההורדה של הלקוח ($U_S = D_R$).

זאת מכיוון שברגע שהבאפר מתחיל להתמלא גודל ה rwnd קטן וגודל החבילות שהספק ישלח קטן יותר ויותר. מהירות ההורדה של הלקוח אינה מושפעת מה Flow Control ולכן כאשר קצב העלאה של הספק יקטן הבאפר יחל להתרוקן יותר ויותר עד שהם יגיעו למעיין שוויון שבו כל כמות המידע שנכנסת לבאפר זהה לכמות המידע שיוצאת ממנו בכל שנייה. ובעבור כמות גדולה מספיק של מידע וזמן מתקיים שההספק יהיה זהה למהירות ההורדה של הלקוח: $P = D_S$.

- כאשר אין את ה Flow Control של TCP, הספק שולח תמיד חבילות גדולות ככל הניתן, ולכן הוא ימשיך לשלוח חבילות מבלי לדעת את כמות המקום הפנוי שנשאר בבאפר.
לצורך הדוגמה נניח ש
$U_S = 4096\ (Bytes/sec),\ D_R = 2048\ (Bytes/Sec),\ Buffer\ Size = 4096\ Bytes$
בכל שנייה שעוברת הספק יכול למלא את הבאפר לגמרי, אך ללקוח לוקח שתי שניות לנקות אותו, מכיוון שהספק תמיד ישלח חבילות גדולות ככל הניתן לבאפר הוא תמיד ישלח חבילות בגודל 4096 בייטים. ולכן כל חבילה שנייה תלך לאיבוד, ומתבזבז הרבה זמן על תשדורת חוזרת של מידע שכבר נשלח לבאפר בעבר. ולכן ההספק יורד.
כלומר כאשר אין Flow Control, ו $U_S > D_R$, מתרחש אובדן גדול של מידע ובזבוז זמן גדול על תשדורת חוזרת של החבילות, ואז ההספק הכולל יורד.

ולכן כאשר יש Flow Control, הביצועים יהיו תלויים אך ורק בלקוח ולא בספק (כיוון שההספק יהיה שווה לקצב ההורדה של הלקוח). בניגוד לכשאין Flow Control, ואז ההספק הכולל היה יורד משמעותית והביצועים היו פחות טובים.

3. במערכות רשת שבהן קיימים מספר מסלולים בין המקור ליעד, תפקיד הניתוב הוא לבחור את הנתיב היעיל ביותר להעברת המידע, תוך איזון בין פרמטרים כמו מהירות, אמינות, אבטחה וניצול משאבי הרשת.
קיימים מספר גורמים אשר משפיעים על בחירת הנתיב, ולפיכך גם משפיעים על ביצועי הרשת[1]:
   - אורך המסלול- ככל שהמסלול ארוך יותר, כך הזמן שלוקח למידע להגיע מהמקור אל היעד גדל.
   לשם כך קיימים מספר אלגוריתמים שיכולים לעזור לנו לבחור את המסלול הקצר ביותר בין המקור ליעד:
   Dijkstra שנועד למצוא את המסלול הקצר ביותר בגרף עם משקלים אי שליליים ו Bellman Ford שנועד למצוא את את המסלול הקצר ביותר בגרף עם משקלים שליליים ללא מעגלים שליליים.
   - עומס הרשת- במצבים שבהם מסלול מסויים עמוס בתעבורה רבה, המידע שהמקור מנסה לשלוח עלול להתעכב יותר, שכן הנתבים לאורך המסלול צריכים לחלק את הטיפול שלהם בין כל זרמי המידע השונים.

[1] חלק מהמידע נלקח מ https://www.youtube.com/watch?v=gQtgtKtvRdo

- זמן השהיה- פרק הזמן שלוקח למידע מהרגע שהוא מתקבל בנתב לאורך המסלול עד לרגע שבו הוא יוצא מהנתב להמשך דרכו במסלול.
- רוחב פס- כמות הנתונים שיכולים לעבור בנתיב בו זמנית, נתיבים עם רוחב פס יותר גבוה מסוגלים להעביר כמותיות גדולות יותר של מידע בפרקי זמן קצרים יותר.
- אבטחת המסלול- זהו מרכיב חיוני בתהליך הבחירה. גם אם נמצא המסלול הקצר ביותר, ייתכן שימצא בו תוקף פוטנציאלי שעלול ליירט או לשנות את המידע המועבר.
  כדי למנוע תרחיש כזה לפעמים יבחר מסלול יותר איטי אך בטוח.
- יתירות ושרידות- כדי להבטיח שהרשת תמשיך לתפקד במקרה של כשל בנתיב במסלול בין המקור ליעד, צריכות להיות ברשת מספר מסלולים חלופיים ביניהם, המאפשרים חילוף מהיר במצב שבו המסלול שנבחר קרס או עמוס יתר על המידה.
  לשם כך קיימים שני סוגים של מנגנוני ניתוב, ניתוב סטטי שנבחר מראש ומשתנה אך ורק באופן ידני או כאשר המסלול הנבחר אינו שמיש. ומסלול דינמי שמשתנה בזמן אמת בהתאם לתנאי הנתבים לאורך המסלול.

4. איך MPTCP משפר את הביצועים של המערכת?

בשביל לראות איך MPTCP משפרת את הביצועים של המערכת נראה ראשית את הבעיות הקיימות ב TCP רגיל[2]:

1. TCP עבודת על נתיב אחד בלבד ללא קשר למבנה הרשת שבה מועבר המידע.
2. TCP קשור לכתובת המקור והמטרה של קצבות המסלול, ולכן כאשר מחליפים רשת (לדוגמה בעת מעבר מחיבור wifi אחד לאחר) כתובת המקור או המטרה יכולה להשתנות וכל החיבורי TCP יופסקו ויצטרכו להתחיל מחדש.
3. כאשר יש מספר חיבורי TCP על אותו המסלול יש התנגשות ביניהם וכל אחד מהחיבורים מנצל רק חלק מיכולת העברת הנתונים של המסלול שבו הם משתמשים.

כעת נראה איך MPTCP פותר כל אחת מהבעיות הללו:

1. MPTCP מאפשר חיבור דרך מספר נתיבים שונים (עבור נתבים שתומכים בו), כך שגם במקרה ובו אחד הנתיבים מפסיק לעבוד או שקיים בו עומס, שאר הנתיבים עובדים וממשיכים להעביר מידע. והפרוטוקול מקטין את כמות המידע שהוא מעביר במסלול הבעייתי או מפסיק להעביר בו מידע באופן מוחלט.
2. MPTCP מאפשרת לפתוח כמה מסלולים בו זמנית, חלקם דרך חיבורים שונים (לדוגמה כמה רשתות wifi שונות), ולכן כאשר מתנתקים מחיבור אחד רק חלק מהמסלולים יתנתקו ויצטרכו להתחיל מחדש, בעוד שהשאר ימשיכו לפעול כרגיל (דורש שימוש במספר ממשקי חיבור אינטרנט).
   שינוי זה מאפשר לשמור על חיבור פעיל בכל רגע נתון והתאוששות מהירה יותר לאחר המעבר[3].
3. ב MPTCP, כאשר מכשיר התומך ב MPTCP, נתקל במסלול בו יש התנגשות עם מידע ממקור אחר, הוא ינסה לפתוח חיבור נוסף, ובכך לחלק את המידע בין המסלולים השונים, ולהפחית את העומס על החלק במסלול בו ישנה התנגשות.
   כך כל אחד מהחיבורים השונים מקבל אחוז גבוה משמעותית מיכולת העברת הנתונים של המסלולים שבהם הם משתמשים.

בכך ש MPTCP מאפשר פתיחה של חיבורים נוספים והקטנת כמויות המידע שהוא מעביר דרך מסלולים עמוסים הוא עובד במקרה הגרוע ביותר עם אותה הקיבולת של חיבור TCP רגיל, אך במקרה הטוב ביותר מאפשר ניצול גבוה משמעותית (ויותר קרוב לקיבולת המקורית) של כל נתב.

---

[2] המידע על MPTCP וההבדלים שלו מ TCP נלקח מ https://www.youtube.com/watch?v=k-5pGlbiB3U
[3] ראו גרף לדוגמה בדקה 40:54 בסרטון.

5. **אנו מנתחים תעבורה ברשת וסמנו לב לאובדן גדול של חבילות בין שני נתבים, ואנו צריכים לחקור את הסיבות הפוטנציאליות לאובדן חבילות בשכבות הרשת והתעבורה.**

i. גורמים פוטנציאליים לאובדן חבילות בשכבת הרשת הם:

- עומס על הנתב: מספר רב של אנשים שולחים או מקבלים מאותו הנתב כמות גדולה של מידע, כך שהנתב לא מצליח לעמוד בקצב העברת המידע, ונוצרת האטה בקצב ולעיתים אף אובדן של חלק מהמידע.
  פתרונות אפשריים לבעיה זו:
  1. החלפת המסלול לנתב פחות עמוס.
  2. אם זהו נתב שנוטה להיות תחת עומס גבוה ניתן לשדרג אותו לנתב איכותי יותר בעל קיבולת יותר גדולה.
  3. הוספת נתבים נוספים לרשת שיוכלו לספק אלטרנטיבה לנתב הבעייתי, ובכך לחלק את העומס.
- תקלה בנתב: נתבים הם מכשירים שלהם יש שני חלקים, החלק החומרתי והחלק התוכנתי (hardware and software), בנתב יכולה להיות תקלה בתוכנה המתוכנת עליו או תקלה פיזית במכשיר אשר יכולות לפגוע בתיפקודו וביכולת שלו להעביר הלאה חבילות.
  פתרונות אפשריים לבעיה זו:
  1. החלפת המסלול לנתב שאינו תקול.
  2. תיקון הנתב או לסירוגין החלפתו.
- קיבולת בנתב: כאשר נתב מסויים מגיע לקיבולת זיכרון מקסימלית, אין לו יכולת לשמור את המידע של חבילות חדשות שנשלחות אליו, ולכן כל חבילה נוספת שתתקבל אצלו תמחק ותאבד.
  פתרונות אפשריים לבעיה זו:
  1. החלפת המסלול לנתב שאינו הגיע לקיבולת מקסימלית.
  2. להאט את קצב שליחת הנתונים לקצב שיאפשר לנתב לפנות מקום בקצב זהה או מהיר יותר מהקצב שבו הוא מקבל חבילות חדשות.
- מגבלה בגודל המידע שנתב מקבל: לנתבים יש מגבלות על גדול החבילות שהם יכולים לקבל ולשלוח, וכאשר נתב מקבל חבילה גדולה מידי, אם יש לו את היכולת – הוא מפרק אותה למספר תתי חבילות בגודל שהוא יכול להתמודד איתן. אך כאשר הוא שולח אותן הלאה הן יכולות להגיע לנתב שלא תומך בתתי חבילות ולא יודע לחבר אותן מחדש, ובכך חלק מהמידע נאבד.
  פתרונות אפשריים לבעיה זו:
  1. מציאת הנתב בעל הקיבולת הקטנה ביותר במסלול ושליחת חבילות בגדלים מתאימים לקיבולת זו.
  2. מציאת הנתבים במסלול שמקבלים תתי חבילות ולא יודעים לחבר אותן בחזרה לחבילה המקורית, או להחליף אותם בנתבים חדשים שכן יכולים לחבר ולפרק את החבילות, או לשנות את המסלול כך שהחבילות לא ישלחו אליהם.
- בעיות תצורה: בעיות האלו נגרמות כתוצאה מהגדרות שגויות בטבלאות הניתוב של הנתב, מה שמוביל להעברת מידע לכתובות שגויות ובכך למידע ללכת לאיבוד, או ע"י שליחתו לכתובת שאינה נכונה בכלל, או ע"י הכנסתו למסלול מעגלי של נתבים.
  פתרון אפשרי לבעיה זו:
  למצוא את הנתבים בהם יש טבלאות ניתוב שגויות ולתקן אותן.
- מרחק גדול בין המקור ליעד: בשכבת הרשת יש את פרוטוקול ה ip שקיים בו השדה TTL, כאשר עם כל קפיצה מנתב לנתב הוא קטן באחד, כאשר

הוא מגיע לאפס הנתב שהחבילה נמצאת אצלו מוחק את החבילה ולא שולח אותה. אם המרחק בין המקור ליעד הינו גדול יותר מה TTL המקורי או שהחבילה עושה מספר גדול של פניות שגויות, ה TTL שלה יכול להגיע לאפס והיא תאבד.

פתרונות אפשריים לבעיה זו:

1. הוספת דרכים חדשות בין המקור ליעד ע"י הוספת נתבים חדשים לרשת אשר יקצרו את המסלול.
2. הגדלת ה TTL של החבילות בשביל לאפשר להם יותר הזדמנויות להגיע אל היעד.
3. בדיקת כמות הקפיצות הנדרשות בשביל להגיע מהמקור אל היעד בעזרת הפונקציות ping ו traceroute.

- נתבים מיושנים: קיימים נתבים מיושנים שלא יודעים להתמודד עם פרוטוקולים חדשים, במידה והם מקבלים חבילות המשתמשות בפרוטוקולים כאלו כמו QUIC, הם מוחקים אותן.

פתרונות אפשריים לבעיה זו:

1. החלפת הנתבים הבעייתים וה"מנוונים" שלא יודעים לתפעל פרוטוקולים חדשים בנתבים חדשים שכן יכולים לעשות זאת.
2. החלפת המסלול הנבחר למסלול הכולל רק נתבים שיודעים להשתמש בפרוטוקולים חדשים.
3. הסוואת הפרוטוקולים החדשים כפרוטוקולים ישנים יותר שנתבים ישנים יותר כן יודעים להתמודד עימם (לדוגמה MPTCP מתנהג כמו TCP בנתבים ישנים שלא תומכים בו).

- התקפות סייבר: התקפות כמו DDoS על הנתבים בדרך יכולות להעמיס על השרתים ולגרום לחבילות האמיתיות שמגיעות אליהם להאבד בין החבילות המזויפות של התוקף.

פתרונות אפשריים לבעיה זו:

1. הימנעות משימוש בנתבים וברשתות שעוברות דרך טריטוריות עוניינות.
2. הימנעות משימוש בנתבים בעלי פרצות אבטחה ידועות.
3. תחזוקת נתבים ישנים עם עדכוני תוכנה וחומרה שיגנו עליהם ממתקפות כאלו ויאפשרו להם להבדיל בין חבילות אמיתיות לחבילות מזויפות של התוקף.
4. שימוש ב Firewalls בנתבים בשביל למנוע מגורם חיצוני להשיג אליהם גישה.

ii. גורמים פוטנציאלים לאובדן חבילות בשכבת התעבורה הם:

- שימוש ב UDP: פרוטוקול UDP הינו פרוטוקול שמתעדף מהירות על פני אמינות ולכן כאשר משתמשים בו יכול להיות אובדן מידע בדרך.

פתרונות אפשריים לפתרון הבעיה:

1. לעבור להשתמש בפרוטוקול .QUIC
2. לעבור להשתמש בפרוטוקול TCP.

- שימוש ב QUIC: לפרוטוקול QUIC אמנם יש שיטות לשמור על אמינות בניגוד ל UDP, אך מכיוון שכל הערכים בו מוצפנים, יכולים להיות נתבים שימחקו חבילות המשתמשות בו.

פתרונות אפשריים לבעיה זו:

1. הימנעות ממעבר בנתבים שלא יודעים להתמודד עם ההצפנה של (יותר פתרון שקשור לשכבת הרשת).
2. לעבור להשתמש בפרוטוקול TCP.

- Congestion: כאשר קצב העברת המידע של הספק גדולה משמעותית מאשר קצב העברת המידע של הלקוח, יכול להתרחש מצב שבו חבילות מתקבלות אצל הלקוח מהר מאשר שהוא יכול להתמודד איתן. בעקבות זאת תגרם האטה בקצב העברת המידע הכולל ואף לאובדן מידע. (בעיה זאת אינה קיימת כאשר משתמשים בפרוטוקול עם מנגנון ל Congestion Control כמו TCP או QUIC).
  פתרונות אפשריים לבעיה זו:
  1. אם לא משתמשים בפרוטוקול שיש לו Congestion Control לעבור להשתמש באחד שיש לו כגון TCP או QUIC.
  2. אם כן משתמשים בפרוטוקול שיש לו Congestion Control ועדיין נתקלים בבעיה של אובדן חבילות, ניתן לעבור לפרוטוקול שמחלק את המידע ומעביר אותו דרך מספר נתבים שונים כגון MPTCP.
- שגיאות רשת: קיימות הפרעות חיצוניות ברשת שיכולות לפגוע באיכות החבילות ולשנות בהן את הביטים בעת המעבר מנתב לנתב, כאשר מתקבלת חבילת TCP עם שגיאה בה (ניתנת לזיהוי בעזרת ה checksum), נשלחת בקשה לשידור חוזר שלה.
  פתרון אפשרי לבעיה זו:
  שימוש בפרוטוקולים שיש להם דרך להתמודד עם זיהוי שגיאות כמו TCP ו QUIC.
- התקפות סייבר: התקפות כמו Man In the Middle, יכולות לגרום לאובדן נתונים כאשר גורם חיצוני נמצא בין המקור ללקוח ו"מיירט" את החבילות בדרך, הן את החבילות עצמן והן את ה ACK-ים עליהן (במידה ומדובר ב TCP) או כאשר הוא משנה את ערכיהן וגורם לאובדן שלהן בהמשך הדרך.
  פתרון אפשרי לבעיה זו:
  שימוש בפרוטוקולים מוצפנים שיקשו על גורם שלישי לשנות את המידע בצורה שאינה תהיה ברורה ליעד כאשר החבילה תגיע אליו לדוגמה QUIC.

חלק 2 קריאת מאמרים:

שם המאמר:
Analyzing HTTPS Encrypted Traffic to Identify
User's Operating System, Browser and Application

שאלה 1)
התרומה העיקרית של המחקר היא הצגת שיטה חדשנית ומקורית, המאפשרת לתוקף לזהות בקלות פרטים על המשתמש מתוך תעבורת HTTPS מוצפנת.

שיטה זו מנצלת תכונות ייחודיות של התנהגות הדפדפנים, כמו ההתנהגות המתפרצת של תעבורת הרשת, וכן מאפיינים של פרוטוקולי SSL ו -TLS .

באמצעות ניתוח נתונים מגוונים, הכוללים מערכות הפעלה שונות, סוגי דפדפנים ואפליקציות שונים, הצליחה השיטה להגיע לדיוק מרשים של למעלה מ-96% בזיהוי.

שאלה 2)

תכונות התעבורה שהמאמר משתמש בהן מחולקת **לשתי רשימות של תכונות**, אחת תכונות בסיסיות והשנייה תכונות החדשות.

רשימת תכונות בסיס:
אלו תכונות נפוצות המשמשות רבות במחקרי סיווג תעבורה קודמים. הן מתמקדות במדדים סטטיסטיים בסיסיים של זרימת המידע, כמו גודל חבילות, תזמון וקצב זרימה.

Forward packets, # Forward total Bytes, Min forward inter-arrival time difference, Max # forward inter-arrival time difference, Mean forward inter-arrival time difference, STD forward inter-arrival time difference, Mean forward packets, STD forward packets, # Backward packets, # Backward total Bytes, Min backward inter-arrival time difference, Max backward inter-arrival time difference, Mean backward inter-arrival time difference, STD backward inter-arrival time difference, Mean backward packets, STD backward packets, Mean forward TTL value, Minimum forward packet, Minimum backward packet, Maximum forward packet, Maximum backward packet, # Total packets, Minimum packet size, Maximum packet size, Mean packet size, Packet size variance

רשימת תכונות חדשות:
אלו תכונות חדשניות שהוצגו לראשונה במאמר זה. הן מנצלות מאפיינים ייחודיים של פרוטוקולי SSL/TLS ונתוני TCP, כמו גם את ההתנהגות המתפרצת של תעבורת דפדפנים. תכונות אלו נועדו לשפר את יכולת הסיווג של התעבורה המוצפנת ולהגיע לרמות דיוק גבוהות יותר.

TCP initial window size, TCP window scaling factor, # SSL compression methods, # SSL extension count, # SSL cipher methods, SSL session ID len, Forward peak MAX throughput, Mean throughput of backward peaks, Max throughput of backward peaks, Backward min peak throughput, Backward STD peak throughput, Forward number of bursts, Backward number of bursts, Forward min peak throughput, Mean throughput of forward peaks,

Forward STD peak throughput, Mean backward peak inter-arrival time diff, Minimum backward peak inter-arrival time diff, Maximum backward peak inter-arrival time diff, STD backward peak inter-arrival time diff, Mean forward peak inter-arrival time diff, Minimum forward peak inter-arrival time diff, Maximum forward peak inter-arrival time diff, STD forward peak inter-arrival time diff, # Keep alive packets, TCP Maximum Segment Size, Forward SSL Version

שאלה 3)

התוצאות העיקריות של המחקר מצביעות על כך שכאשר משתמשים רק בתכונות הבסיסיות או רק בתכונות החדשות, מתקבל דיוק גבוה של 93.52% בכל אחת מהבדיקות. עם זאת, כאשר משלבים את שתי קבוצות התכונות יחד, הדיוק עולה ל-96.06%, שיפור של 2.54% בהשוואה לשימוש בתכונות בנפרד.

באמצעות השימוש המשולב בתכונות (כלומר גם תכונות בסיסיות וגם החדשות), הסיווג עבור רוב ה-tuples היה כמעט מושלם, כאשר כמעט ולא נרשמו מקרים חריגים של בלבול בין קטגוריות דומות או בין מחלקות לא ידועות.

התובנות העיקריות מהתוצאות:

השפעת שילוב התכונות: התוספת של תכונות חדשות, בעיקר אלו הקשורות לפרוטוקולי SSL/TLS ולהתנהגות המתפרצת של הדפדפן, תרמה לשיפור משמעותי ביכולת הזיהוי.

פוטנציאל זיהוי למרות הצפנה: הממצאים מראים כי גם כאשר התעבורה מוצפנת בפרוטוקולים מאובטחים כמו HTTPS, עדיין ניתן לחשוף מידע רגיש על מערכת ההפעלה, הדפדפן אפליקציה של המשתמש.

השלכות אבטחת מידע: תוקף פוטנציאלי יכול לנצל את המידע הזה על מנת לבצע התקפות ממוקדות או לאסוף נתונים סטטיסטיים על קבוצות משתמשים, מה שממחיש את חולשת מערכות אבטחה הקיימות.

שם המאמר:
Early Traffic Classification with Encrypted ClientHello: A Multi-Country Study

שאלה 1)
התרומות העיקריות של המחקר הינן התרומות הבאות:

- איסוף מאגר מידע בעבור eTC (early Traffic Classification), שמכיל יותר מ 600,000 זרמי TLS (Transport Layer Security) המחולקות ל-19 קטגוריות מדוייקות.
  מאגר זה הינו מגוון בסוגי הפרוטוקולים, בזמנים, בגיאוגרפיה והמכשירים שיצרו את זרמי ה TLS.
  מאגר זה הינו ייחודי ברשת, ואין שני לו אשר מגוון כמוהו בכל הקטגוריות המוזכרות לעיל.
- פיתוח אלגוריתם חדש לקלסיפיקציה של תעבורה מוצפנת (ETC) בשם: hybrid Random Forest Traffic Classifier (hRFTC).
- בדיקה של יכולת ההתמודדות של אלגוריתמים חדשניים מסוג eTC עם ההטרוגניות של מידע במקרה של (Encrypted ClientHello) ECH וכמה הם מצליחים להכליל מכמות קטנה שלו.
- הדגמה שגם בעבור האלגוריתמים ההיברידיים הטובים ביותר של TC שאומנו באזור גיאוגרפי אחד, יהיה צורך לאמנם מחדש באזור אחר, בעקבות הבדלים משמעותיים בתבניות התעבורה.

שאלה 2)
תכונות התעבורה שהמאמר משתמש מתחלקות לשני סוגים:

1. תכונות מבוססות חבילה:
   - TLS Cipher Suite (for both CH and SH)
   - TLS Cipher Suite Length (for both CH and SH)
   - TLS Key Share Group (for both CH and SH)
   - TLS Extensions (for both CH and SH)
   - Recomposed CH and SH
2. תכונות מבוססות זרימה:
   - Number of Packets until first DL packet with application data - Count of UL and DL with non-zero L4 payload.
   - Packet Size Stats - Mean, STD, min, max, 25th, 50th, 75th percentile, and sum of Packet Sizes - for both UL and DL.
   - Packet Size Pattern - Aligned Vector for the first 6 Packet Sizes - for both UL and DL.
   - Packet Size Unique - Aligned Vector for the 6 largest unique Packet Sizes.
   - Packet Size Histogram - Vector of Packet Size frequencies for bins of 250 bytes - for both UL and DL.
   - IPT Stats - Mean, STD, min, max, 25th, 50th, 75th percentile, and sum of IPTs - for both UL and DL.

תכונות התעבורה החדשות שבהן המאמר השתמש הן:

- קריטריון בחירת החבילות החדש של האלגוריתם (הלוקח חבילות עד חבילת ה DL הראשונה שיש לה application data).
- האלגוריתם כלל תכונות חדשות מבוססות זרימה המיועד לזיהוי מוקדם של תעבורה מוצפנת, הנלקחות בנפרד מחבילות UL ו DL, בצורה המקטינה את העיכוב בקלסיפיקציה שקיים באלגוריתמים מבוססי זרימה.
- הרחבת האלגוריתם RB-RF על מנת שיעבוד גם עם QUIC, ע"י הגדרת זרם להיות רצף חבילות שחולקות את אותו ה QUIC connection ID. הרחבה זאת הייתה נדרשת שכן בפרוטוקול QUIC כמעט כל המידע מוצפן בשביל להימנע משינוי המידע ע"י middle boxes.

בעבור שאלה זו השתמשנו ב chatgpt עם ה prompt:
"What traffic features does the paper use, and which are novel?"

שאלה 3)

המאמר הניב מספר תוצאות, המרכזיות ביניהן הן:

1. **התוצאה:**

TABLE 11. **Full dataset per class F-score for different classifiers.**

Class	F-score [%]						
	Hybrid Classifiers			Flow-based Classifier	Packet-based Classifiers		
	hRFTC [proposed]	**UW** [35]	**hC4.5** [34]	**CESNET** [63]	**RB-RF** [24]	**MATEC** [33]	**BGRUA** [32]
BA-AppleMusic	**92.1**	89.5	80.2	89.2	25.5	13.1	14.5
BA-SoundCloud	**99.6**	98.9	97.8	98.7	84.4	81.8	82.0
BA-Spotify	**93.6**	90.8	89.0	88.5	16.3	0.0	3.6
BA-VkMusic	**95.7**	89.7	88.5	91.8	2.6	2.1	3.2
BA-YandexMusic	**98.5**	93.2	93.7	92.5	1.8	0.2	0.1
LV-Facebook	**100.0**	99.7	99.8	99.8	**100.0**	**100.0**	**100.0**
LV-YouTube	**100.0**	**100.0**	99.9	**100.0**	**100.0**	99.0	98.4
SBV-Instagram	**89.7**	74.7	76.5	78.8	10.0	6.3	6.4
SBV-TikTok	**93.3**	81.8	81.8	76.3	38.3	34.3	34.5
SBV-VkClips	**95.7**	94.0	91.3	92.4	53.2	37.7	46.0
SBV-YouTube	**98.2**	96.6	94.7	96.4	1.1	0.2	0.2
BV-Facebook	**87.7**	78.2	79.7	77.6	5.6	3.2	3.8
BV-Kinopoisk	**94.1**	84.1	85.8	89.8	5.4	4.0	4.1
BV-Netflix	**98.5**	97.2	95.2	93.7	50.7	52.3	56.1
BV-PrimeVideo	**91.3**	86.7	84.1	84.7	32.5	24.7	26.8
BV-Vimeo	**94.8**	90.5	90.2	81.4	72.0	19.5	68.6
BV-VkVideo	**88.6**	80.5	80.4	79.7	10.5	0.0	0.1
BV-YouTube	**85.9**	84.3	77.0	78.5	22.3	19.6	20.2
Web (known)	**99.7**	99.5	99.4	99.4	98.0	98.0	98.0
Macro-F-score (average)	**94.6**	89.9	88.7	88.9	38.4	31.4	35.1

LV is Live Video, (S)BV is (Short) Buffered Video, and BA is Buffered Audio.

האלגוריתם המוצע במאמר - hRFTC, עובד בצורה יותר טובה מכל האלגוריתמים מבוססי החבילות הטובים ביותר:

- RB-RF - ב 56.3%.
- MATEC - ב 63.3%.
- BGRUA - ב 59.6%.

מבחינת ה F-score שלו.

בנוסף לכך, hRFTC מקטין במחצית את אחוזי השגיאה של אלגוריתמי הקלסיפיקציה הטובים ביותר.

המסקנה:

קיימת יעילות גבוהה בשימוש בקבוצת התכונות המבוססות זרימה וחבילה שהוצעו לשימוש באלגוריתם ביחס לאלגוריתמים הקיימים האחרים.

2. **התוצאה:**

האלגוריתם UW הוציא תוצאות TC פחות טובות באופן משמעותי מאשר התוצאות במאמר המקורי שלו, אשר יחסו לעובדה שברוב המאמרים על ETC רק מחביאים מאלגוריתם ה TC את ערך ה SNI ב CH אך לא מחביאים את אורכו, דבר שכן מוסתר ב ECH בעזרת padding.

בשביל להבטיח eTC, שקלו פחות חבילות לתכונות מבוססות זרימה, דבר שיכל להוביל פגיעה ביכולות של האלגוריתם CESNET, המבוסס על זרימה.

המסקנה:

התוצאות של CESNET הנמוכות יותר משל hRFTC ו UW, מאשרות שהמטא דאטא של TLS שימושיות בעבור TC, ואלגוריתמים היברידיים של eTC יכולים להשיג תוצאות TC בעלות איכות גבוהה יותר מאלגוריתמים המבוססים על זרימה.

3. **התוצאה:**

TABLE 13. hRFTC: Sum of GI values over payload, PS, and IPT features normalized by the sum of all GI values.

Payload		IPT	PS			
CH	SH	IPT Stats	PS Hist	PSs Unique	PS Stats	PS Pattern
0.18	0.09	0.16	0.05	0.11	0.16	0.26
0.27		0.16	0.57			

התכונות הקשורות לגודל החבילה הן החשובות ביותר בזיהוי, כאשר הן מהוות מעל ל-50% מסך כל החשיבות.
בנוסף, למרות הניסיון להצפין כמה שיותר מידע בהודעות TLS בעזרת ETC ואמצעים אחרים, תכונות ה Payload של החבילות מהוות כמעט 30% מהתכונות החשובות לתהליך הזיהוי.
אם זאת ה F-score הנמוכים ביותר שהתקבלו היו בעבור תעבורת QUIC שנוצרה ע"י BV-Youtube, BV-Facebook, and SBV-Instegram.

המסקנה:
בעקבות QUIC Padding, לכל החבילות מהסוג QUIC handshake יש את אותו האורך, ומכיוון שתכונות ה Payload מהוות בערך 30% מהתכונות החשובות בתהליך הזיהוי QUIC PADDING frames פוגעות בתהליך הזיהוי, גם אם אך ורק באופן מוגבל.

4. התוצאה:

הקטנת כמות חומר האימון בעבור האלגוריתם מ 70% ל 10% הובילה להורדה ביכולת הזיהוי של רק 7% (ה F-score קטן רק ב 7).
בנוסף לכך שכאשר hRFTC אומן על רק 10% מהמידע, יכולת הזיהוי שלו הייתה דומה לשל hC4.5 עם אימון של 70% ו UW עם אימון של 40% מהמידע במאגר.
בנוסף לכך, נבדקה היכולת של כל אלגוריתם לתפקד במדינות שונות (טבלה 14).

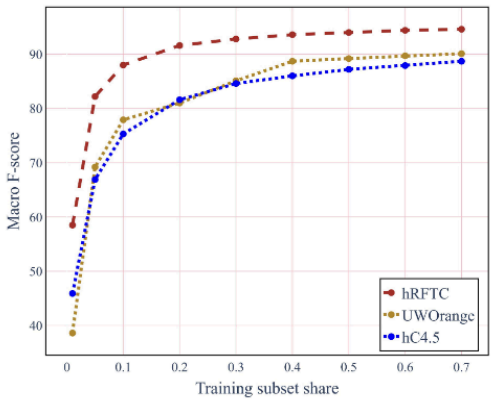


FIGURE 4. F-score depending on the training subset share.

TABLE 14. TC quality depending on training locations.

Test Country	Share in Dataset	Training Country	Classifier Macro F-score [%]		
			hRFTC	hC4.5	UW
Germany	18.8%	Others	38.4	26.9	19.5
Kazakhstan	3.0%	Others	57.3	32.3	27.5
Russia	29.2%	Others	49.8	35.6	20.9
Spain	16.3%	Others	38.5	34.4	12.6
Turkey	25.2%	Others	35.1	26.0	16.4
USA	7.5%	Others	49.2	41.4	21.3

מסקנות:
למרות יכולת ההכלה היוצאת מן הכלל של hRFTC, הקלסיפיקטור אינו מתפקד טוב באיזור גיאוגרפי חדש.
על אף שבכל אחד מהאזורים שנבדקו כמות המידע לאימון הייתה גדולה מספיק בשביל שאלגוריתמים ילמדו במהלך האימון, היא לא עזרה להם להשיג איכות TC טובה בעבור אזורים לא מוכרים.
מה שמראה שקלסיפיקטורים הם אפקטיביים בתוך האיזורים גיאוגרפיים ידועים, אך מתקשים באזורים חדשים.
ובנוסף, למרות שחבילות IPT תלויות מקום, ניתן לראות שתכונות שקשורות לגודל החבילות משתנות גם הן ממקום למקום.
ולכן בשביל להשיג תוצאות קלסיפיקציה באיכות גבוהה, אלגוריתמי eTC מודרניים המבוססים על תכונות של זרימה צריכים להיות מאומנים באזור זהה לאזור שבו ישתמשו בהם.

שם המאמר:

FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition

שאלה 1)

התרומות העיקריות של המאמר:
הן להציג שיטה חדשה של זיהוי תעבורה באינטרנט מוצפנת בעזרת שימוש ב- **FlowPic**, שהיא ממירה נתוני זרימה לתמונות, בעזרת שימוש למידה עמוקה .**CNN**
FlowPic מצליחה **להשיג דיוק מאוד גבוה** של בשימוש לזיהוי קטגוריות תעבורה. בנוסף FlowPic **מתמודדת עם האתגרים והמכשולים** של אמצעי הצפנה בתעבורת הרשת **כמו שימוש ב- VPN** עם דיוק של **99.2%, ו- TOR עם דיוק של 89%.**
בנוסף מסוגל להתמודד עם תעבורה מוצפנת עם דיוק של **78.9% ועד 99.4%** כך שהממוצע דיוק של תעבורה **מוצפנת** היא **89.15%.**
לעומת זאת לשיטות זיהוי קודמות, שימוש ב- **FlowPic** אינו תלוי בשום מאפיינים ידניים וכך מאפשר זיהוי אפליקציות חדשות שלא היה לו ניסיון בעבר עם **דיוק מדהים של 99.9%.**
השימוש שומר על פרטיות בגלל שהוא **אינו זקוק למידע שנמצא בתוך ה- Payload** שנמצא בתוך החבילות, לכן השיטה זאת **חסכונית במשאבים** בשימוש בזמן אמת, מצליחה לעמוד באתגרים בעידן המודרני תעבורה מוצפנת.

שאלה 2)

תכונות שבהן השתמש המאמר:

Packet Size– בודק את גודל כל חבילת נתונים הנשלחת ברשת.

Packet Arrival Time – מחשב מתי כל חבילה מתקבלת ומשתמש בזה כדי לייצר דפוסי תנועה.

Unidirectional Flow Analysis – מסתכל רק על חבילות שנשלחות בכיוון אחד, בלי לבדוק תגובות חזרה.

Payload Size Distribution– בודק איך משתנה גודל החבילות במהלך הזרימה.

Inter-Packet Time – מודד כמה זמן עובר בין חבילה לחבילה.

Flow Duration– מחשב כמה זמן נמשכת הזרימה מהרגע שהתחילה עד שנגמרה.

Packet Frequency– בודק כמה חבילות נשלחות בפרקי זמן מסוימים.

Packet Burst Patterns– מזהה רגעים שבהם יש שליחה אינטנסיבית של חבילות לעומת רגעים של תנועה נמוכה.

Histogram Representation – ממיר את גודל וזמן הגעת החבילות היסטוגרמה דו-מימדית המציגה את תנועת הרשת.

תכונות חדשות שבהן השתמש המאמר:

Flow Pic– במקום לנתח את הנתונים בטבלה, המאמר ממיר את הזרימה תמונה שמייצגת את הדפוסים שלה.

Deep Learning-Based Classification – משתמש ברשתות נוירונים קונבנציונליות (CNN) כדי לזהות דפוסים בתמונה של הזרימה.

Time-Size Matrix– הופך את הנתונים של הגודל והזמן של כל חבילה למערך דו-מימדי שניתן לנתח אותו כמו תמונה.

Pattern Recognition in Traffic – במקום לבדוק פרטים טכניים של חבילות, המערכת מזהה צורות ודפוסים ייחודיים בתנועה.

No Need for Manual Feature Selection (אין צורך בבחירת מאפיינים ידנית) – בניגוד לשיטות קודמות, השיטה אוספת את כל הנתונים הרלוונטיים ומניחה לרשת הנוירונים להבין מהם המאפיינים החשובים.

Robust Encryption Handling (התמודדות עם הצפנה) – מסוגל לנתח תנועה גם אם היא מוצפנת, כי הוא מסתמך רק על דפוסים של גודל וזמן במקום על התוכן של החבילות.

Single Model for Multiple Tasks (מודל אחד למגוון משימות) – אותו מודל משמש גם לזיהוי סוגי תנועה שונים וגם לזיהוי אפליקציות, ללא צורך לשנות את המבנה שלו.

השיטה הזו מאפשרת למערכת לזהות תנועה בצורה מדויקת יותר, גם כאשר יש הצפנה או כש-לא ידוע מראש לאיזו אפליקציה שייכת התנועה.

שאלה 3)

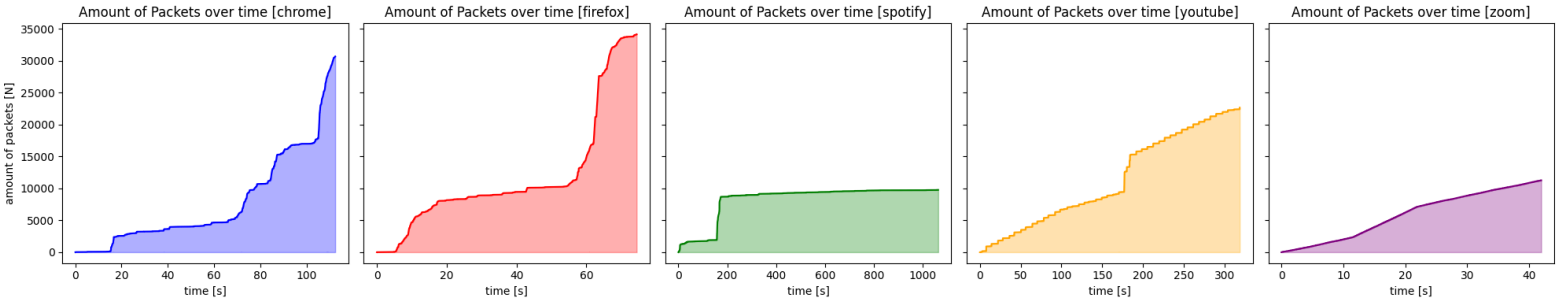
המאמר מציג תוצאות שמראות כי השיטה שהוצגה מצליחה לזהות ולסווג תעבורת רשת בדיוק גבוה, גם כאשר התנועה מוצפנת. עבור תנועה לא מוצפנת **(Non-VPN)**, המודל השיג דיוק של **85.0%**, ואילו עבור תנועה שעוברת דרך VPN, הדיוק עלה ל-98.4%. כאשר התנועה עברה דרך Tor, **הביצועים ירדו ל-67.8%,** מה שמראה הצפנה חזקה יותר מקשה על הסיווג. עם זאת, גם כאשר המודל אומן רק על תנועה לא מוצפנת, הוא **הצליח לזהות בין 78.9% ל-99.4%** מהתעבורה שעוברת דרך **VPN**, דבר המעיד על היכולת שלו לזהות דפוסים כלליים. בניתוח של אפליקציות ספציפיות כמו **Facebook**, **YouTube**, **Skype**, המודל הצליח לזהות אותן **בדיוק גבוה מאוד של 99.7%,** מה שמראה שלכל אפליקציה יש דפוס תנועה ייחודי שניתן לזהות **באמצעות FlowPic.**

מהתוצאות אפשר ללמוד **שהשיטה מצליחה להתמודד עם הצפנה** על ידי זיהוי דפוסים של **גודל וזמן הגעת חבילות**, **במקום להסתמך על תוכן החבילות עצמן**. העובדה שבמודל **יכול להבחין** בין **אפליקציות שונות** כמעט ללא טעות מראה שיש לכל אחת מהן מבנה זרימה אופייני שניתן ללמוד. עם זאת, תעבורה שעוברת דרך Tor **מהווה אתגר משמעותי יותר,** שכן ההצפנה החזקה שלו מקשה על זיהוי הדפוסים. בנוסף, ה**שימוש בגודל ובזמן הגעת החבילות בלבד** הופך את השיטה **המהירה ויעילה**, מה שמאפשר לה לפעול גם בזמן אמת ולדרוש **פחות משאבי חישוב בהשוואה לשיטות קודמות**.

חלק 3:

.3

צילומים של כל אחד מהגרפים שיצרנו:



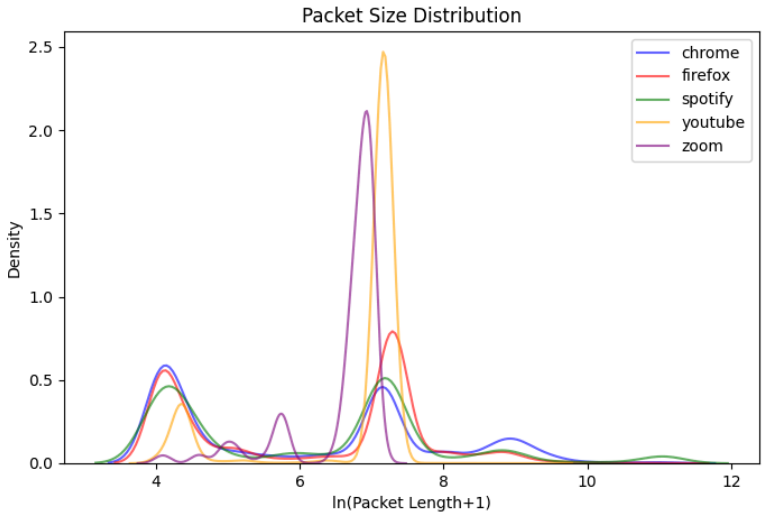
א.

מהגרפים שנוצרו, ניתן להבחין כי קצב הגידול של כמות החבילות משתנה בין היישומים.
היישום עם קצב הגידול הגבוה ביותר הוא **Firefox**, ואחריו **Google Chrome**. לאחריהם, בקצב מתון יותר, ניתן לראות את **Zoom**, ולאחריו **YouTube**. לבסוף, היישום עם קצב הגידול האיטי ביותר הוא **Spotify**

הסדר המדויק של קצב גידול כמות החבילות (מהיר ביותר לאיטי ביותר) הוא:

1. Firefox
2. Google Chrome
3. Zoom
4. Youtube
5. Spotify

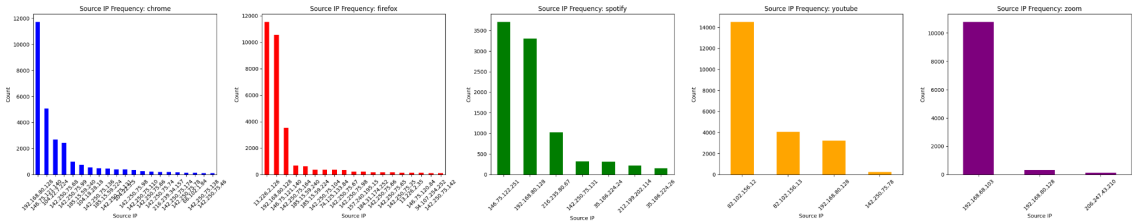
ב.



ניתן לראות שלכל ההקלטות יש אחוז צפיפות גבוה של ערכי $ln($גודל$)$ של החבילות סביב הערכים **4 או 7**.

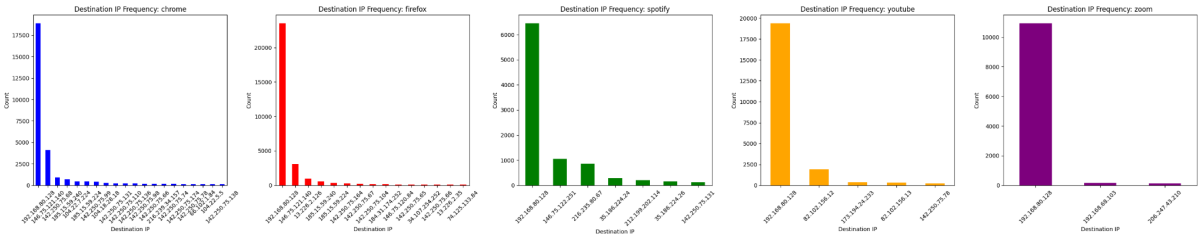
משמעות הדבר היא שרוב החבילות שנשלחו או התקבלו על ידי היישומים השונים מתרכזות בשני תחומים אלו, מה שעשוי להעיד על דפוסי שימוש או מבני נתונים נפוצים בפרוטוקולים השונים.

ג.



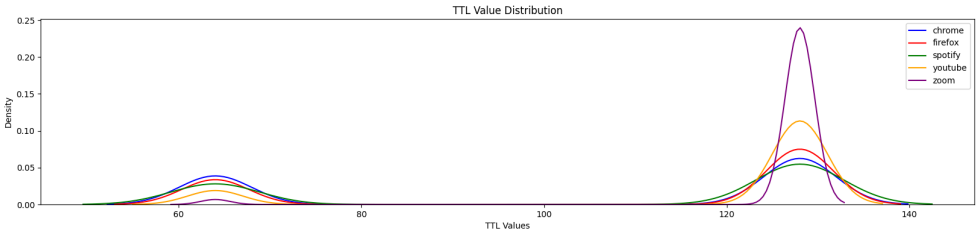
מהגרפים ניתן לראות כי **הדפדפנים** (**Chrome** ו-**Firefox**) יוצרים **תעבורה למגוון רחב יותר של כתובות יעד ושל פורטי יעד**, בהשוואה לאפליקציות ייעודיות כמו **Zoom** ו-**YouTube**. בעוד שהאפליקציות התמקדות בעיקר בשרתים קבועים במספר מצומצם של פורטים ייעודיים, הדפדפנים מציגים פיזור רחב יותר, ככל הנראה בשל הגישה למגוון אתרים ושירותים שונים.

ד.



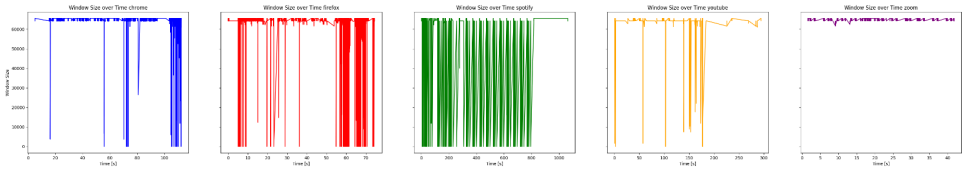
מהגרפים ניתן לראות כי **הדפדפנים** (**Chrome** ו-**Firefox**) יוצרים תעבורה למספר רב יותר של **כתובות יעד** בהשוואה לאפליקציות ייעודיות כמו **Zoom** ו-**YouTube**. בעוד שהאפליקציות התמקדות בעיקר בתקשורת עם מספר קטן של שרתים מרכזיים, הדפדפנים ניגשים למגוון רחב יותר של כתובות, מה שעשוי להעיד על דפוסי שימוש מגוונים יותר התחברות למספר רב של שירותים ואתרים.

ה.



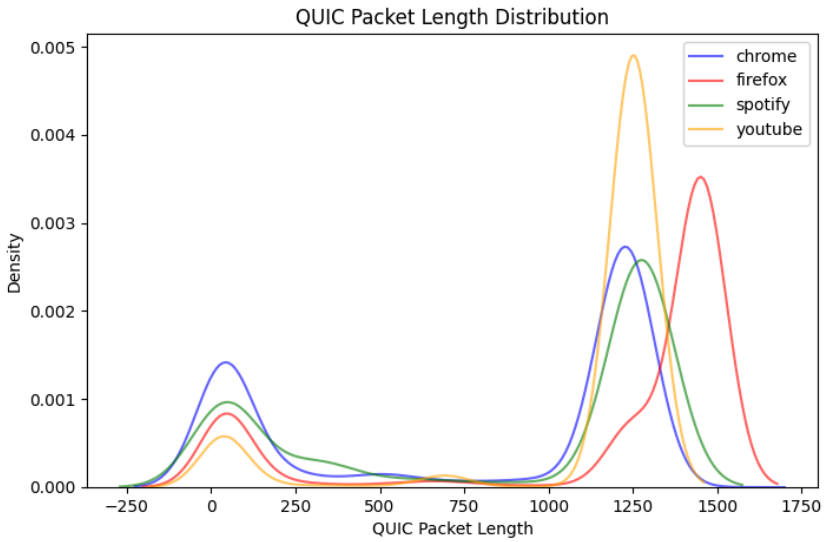
ניתן לראות כי בכל ההקלטות ערכי ה-**TTL** מרוכזים בעיקר סביב **65** ו-**125**. הדבר עשוי להעיד על חבילות שמגיעות משני סוגים שונים של רשתות או מרחקים שונים בין מקור ליעד.

ו.



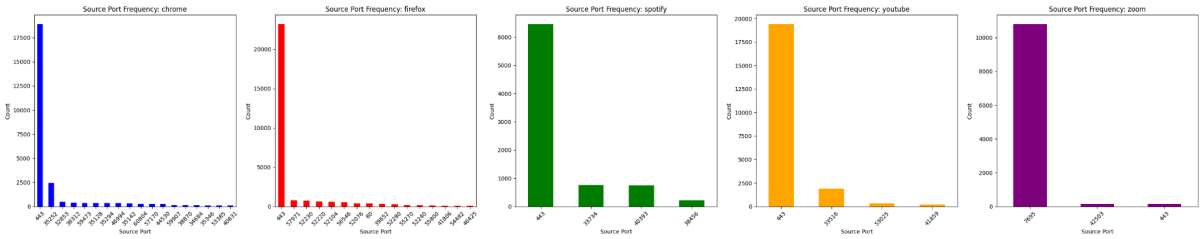
ניתן לראות כי ברוב הזמן **גודל החלון (Window Size) נשמר קרוב למקסימום**, למעט ירידות חדות שמופיעות מדי פעם. ירידות אלו עשויות לקרות עקב עומסים ברשת או מנגנוני בקרה בתעבורה.

ז.



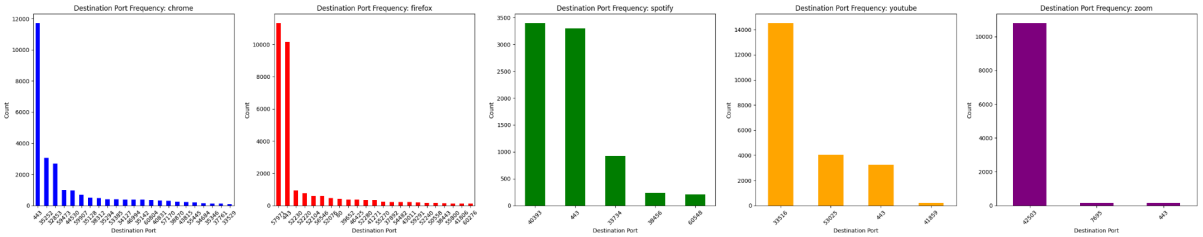
ניתן לראות כל ההקלטות יש שימוש משמעותי **בחבילות QUIC** עם גדלים סביב **1300 בייטים ו-20 בייטים**, מה שמעיד על דפוסי תעבורה אופייניים לפרוטוקול. לעומת זאת, **Zoom אינה משתמשת ב-QUIC כלל**, ולכן אינה מופיעה בגרף.

ח.



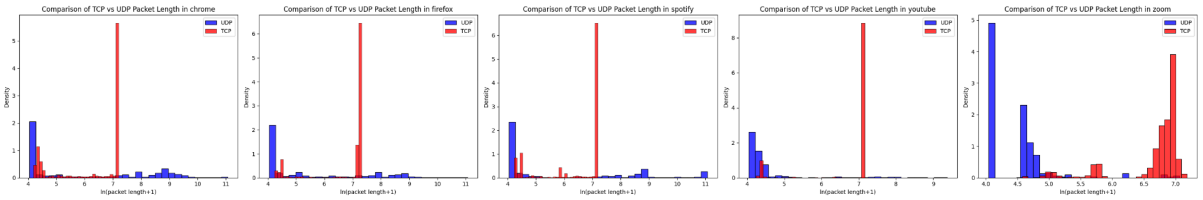
ניתן לראות כי בהקלטות של הדפדפנים יש תעבורה הנשלחת **ממספר רב** יותר של ports ביחס להקלטות של אפליקציות ספציפיות כמו זום או יוטיוב.
Port 443 נמצא בשימוש בכולן.

ט.



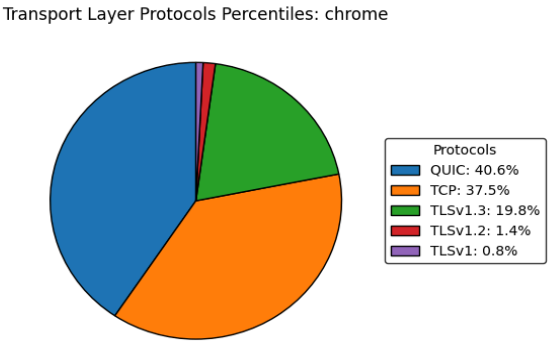
ניתן לראות כי בהקלטות של הדפדפנים יש תעבורה הנשלחת **למספר רב** יותר של ports ביחס להקלטות של אפליקציות ספציפיות כמו זום או יוטיוב.
Port 443 נמצא בשימוש בכולן.

י.

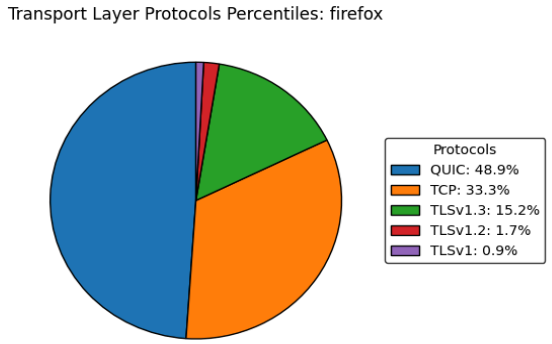


ניתן לראות שבכל ההקלטות יש שימוש נרחב בחבילות TCP ב $ln($גודל$)$ ששווה ל 7.
לעומת זאת ניתן לראות אפליקציית **ZOOM**, משתמשת גם ב- TCP וגם ב-UDP יותר מדפדפונים ואפליקציות אחרות.

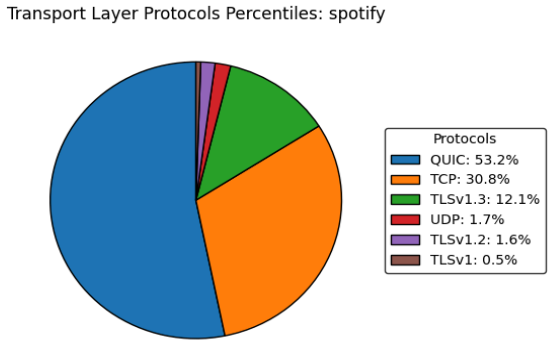
יא.
ב**Chrome**, פרוטוקולי **QUIC** ו-**TCP** תופסים **כמעט 80%** מהתעבורה, עם חלוקה כמעט שווה ביניהם.



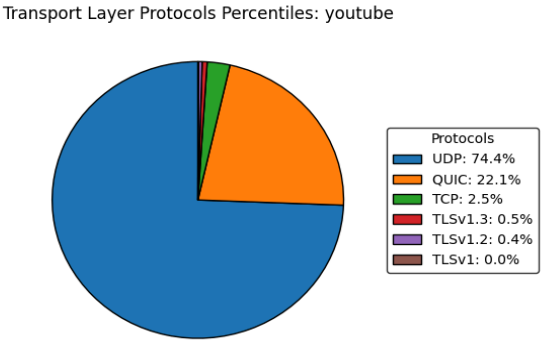
ב**Firefox**, שני הפרוטוקולים יחדיו מהווים **כמעט 70%** מהתעבורה, כאשר **QUIC לבדו מהווה קרוב ל-50%** מסך כל התעבורה.



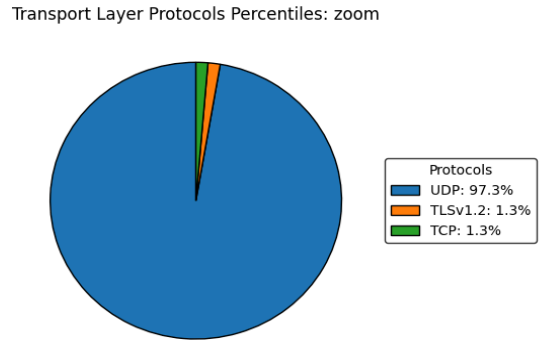
ב**QUIC**, **Spotify ו-TCP** אחראים **ליותר מ-80%** מהתעבורה, כאשר **QUIC לבדו מהווה מעל למחצית** מסך כל התעבורה.



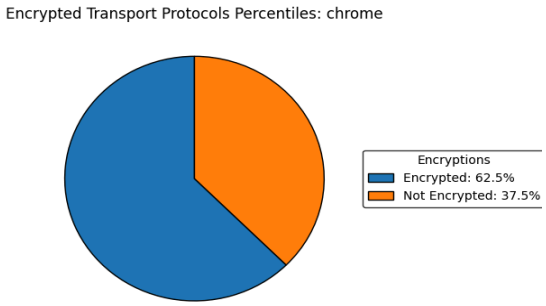
ב**QUIC**, **YouTube תופס כמעט 75%** מהתעבורה, ובשילוב עם **TCP** הם מהווים **כמעט 100%** מהתעבורה בהקלטה.



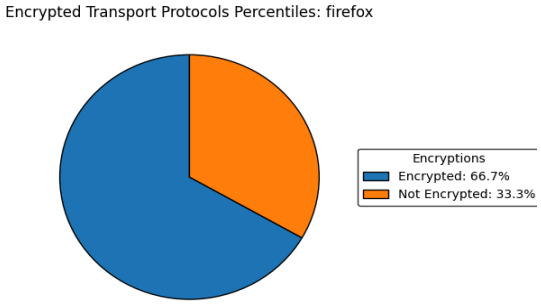
ב**Zoom**, **כמעט כל התעבורה מתבצעת באמצעות UDP**, עם שימוש מזערי ב-TLS וב-TCP.
ב**כל ההקלטות**, ניתן לראות כי **שימוש בפרוטוקולי TLS מהווה לכל היותר 25% מסך כל התעבורה**.



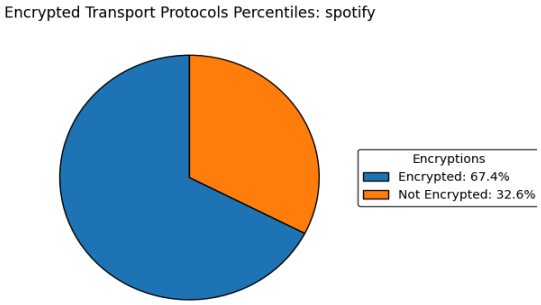
יב.



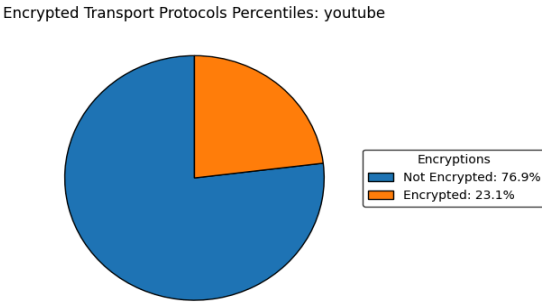
Google Chrome רוב התעבורה נעשית על פרוטוקולים **מוצפנים** באופן אוטומטי (**QUIC** ו **TLS**).



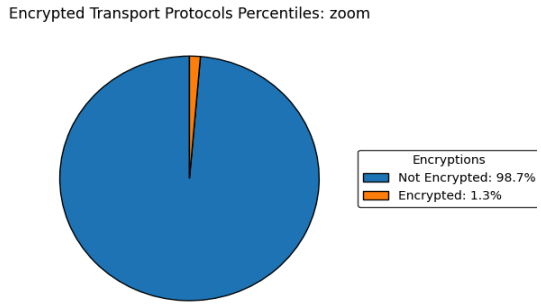
ב-**Firefox**, רוב התעבורה מבוססת על **QUIC ו-TLS**, שהם **פרוטוקולים מוצפנים כברירת מחדל**, מה שמבטיח תקשורת מאובטחת.



ב- **Spotify** רוב התעבורה נעשית על פרוטוקולים **מוצפנים** באופן אוטומטי (**QUIC** ו **TLS**).



ב-**Youtube** רוב התעבורה נעשית על פרוטוקולים **שאינם מוצפנים** באופן אוטומטי (**TCP** ו **UDP**).



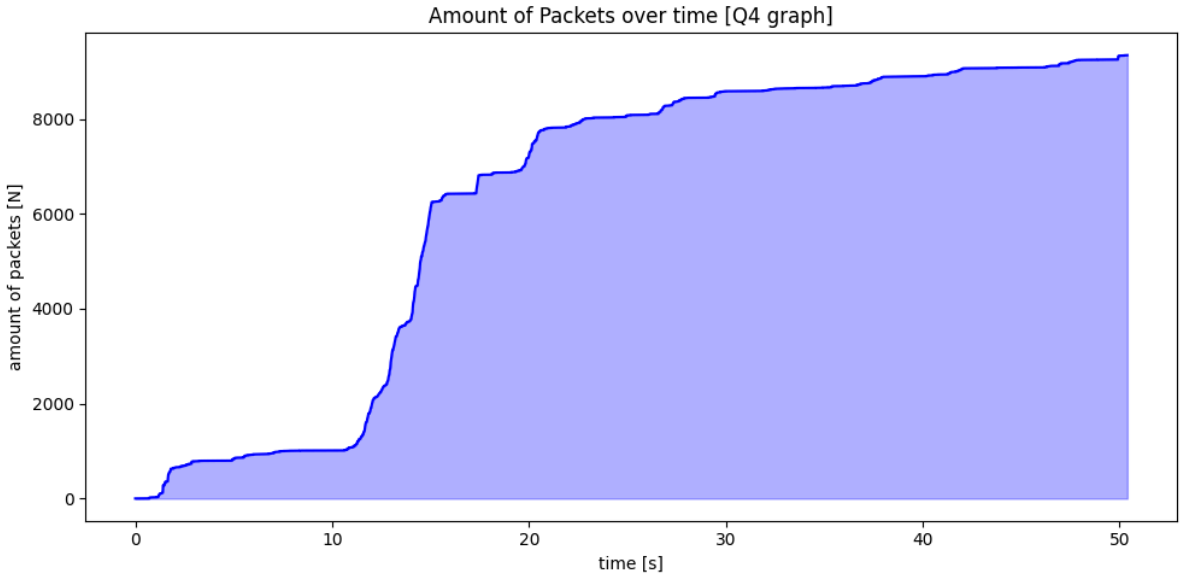
ב- **Zoom** כמעט כל התעבורה נעשית על פרוטוקולים **שאינם מוצפנים** באופן אוטומטי (**TCP** ו **UDP**).

.4
בעבור שאלה זו הקלטנו תעבורה על המחשב והוצאנו ממנה את המידע הבא:

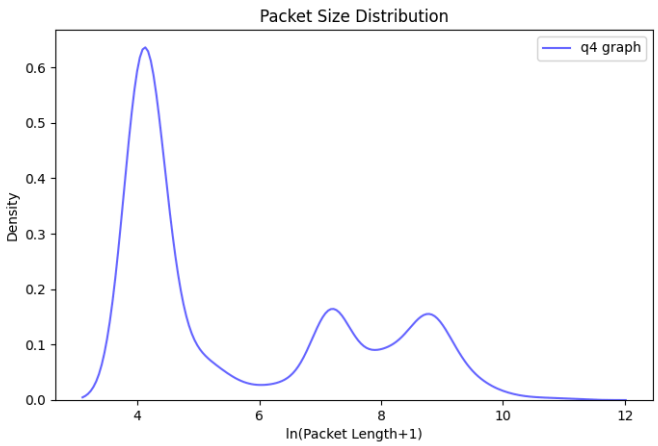
- גודל החבילות
- ה timestamp שלהן
- Hash של ה Tuple של 4 הערכים הבאים:
  - Source IP
  - Destination IP
  - Source Port
  - Destination Port

<u>הערה</u>: הקלטנו תעבורה שנעשת על firefox בחיפוש בדפדפן.

1. נתייחס כעת למקרה שבו לתוקף יש גישה רק לשני הערכים הראשונים:
ברשות התקוף היכולת לראות את כמה חבילות עברו בפרק זמן מסויים ואת גודל החבילות ולכן הוא יכול ליצור מספר גרפים ולהשוואת אותם לתעבורה ידועה:



עם הגרף הנ"ל המייצג את כמות הפקטות שנלקטו לאורך זמן התוקף יכול להשוות את צורתו לצורתם של הגרפים של אפליקציות ואתרים נפוצים ובכך לראות שצורת הגרף מאזכרת את הגרפים של הקלטות הדפדפנים.
בנוסף עם הגרף הבא:

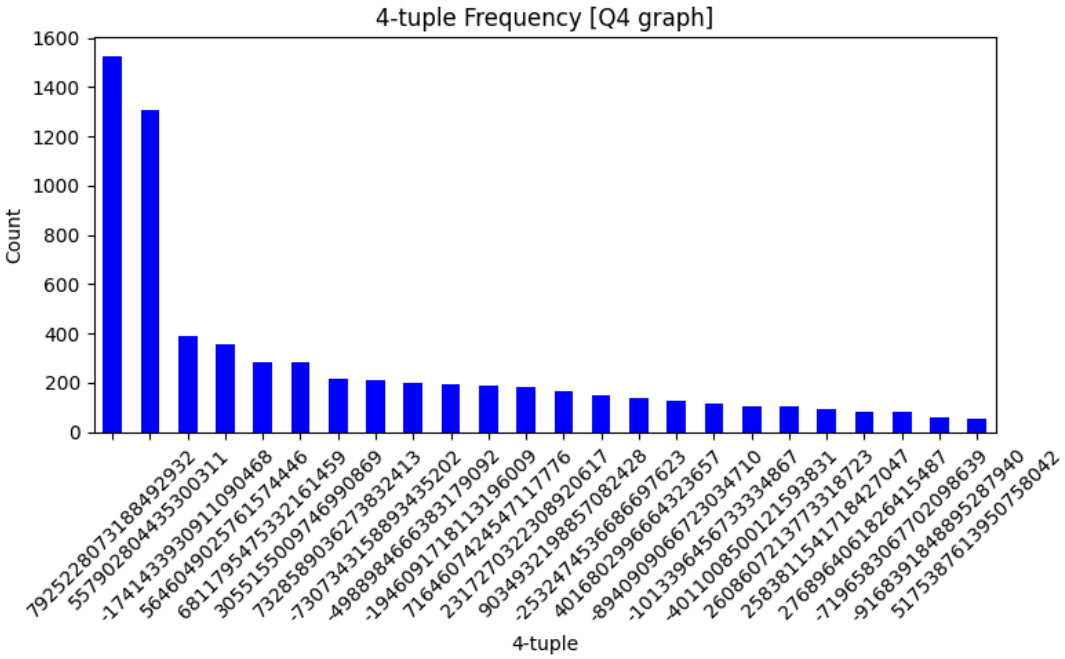


התוקף יכול לזהות שצורתו מזכירה את צורותיהם של הגרפים של דפדפנים ושל ספוטיפיי ובכך לצמצם את התעבורה את אפשרויותיו עוד יותר.

בשביל למנוע את המתקפה במקרה כזה שבו כל שאר המידע מוצפן נוכל לעשות את הדברים הבאים:

- להוסיף padding לכל החבילות בשביל לייצר להן גודל אחיד כך שהתפלגות גדלי החבילות לא תוכל לייצג כלום.
- להכניס רעש למערכת ע"י הוספת חבילות דמה שאינן מיודעות למטרה האמיתית שאליה אנו שולחים או שממנה אנו מקבלים חבילות. בכמות כזאת כך שסך כל החבילות שנשלחו עד אותו הרגע תהיה לא צפוייה רנדומלית, ובכך להסתיר את כמות החבילות האמיתיות שנשלחו ולהסתיר את צורת הגרף.

2. נתייחס כעת למקרה ולתוקף יש גם את ה four-tuple המוזכר לעיל:
בנוסף לגישה לערכים שהיו לתוקף קודם, כעת יש לו גם גישה לערך ייחודי המייצג את התקשורת הייחודית הנעשית בין הלקוח לשרת.
נסתכל על גרף עמודות המראה את תדירות השימושים בכל four-tuple:



בעזרת הגרף הזה התוקף יכול לזהות שעיקר התקשורת נעשית בעזרת שני tuples,
ולהסיק מהם שאחד הוא הtuple שהמקור שלו הוא הלקוח והשני הוא האחד שהמקור שלו הוא השרת.
ובעזרת למידת מכונה הוא יכול אף לדלות מידע על מיהו השרת והports שבהם נעשה שימוש שכן חלק מהמידע נשמר בעת תהליך ה hashing וניתן לגלות אותו עם כמות מספיק גדולה של מידע לאימון ובעזרת אלגוריתם איכותי מספיק.

בשביל למנוע מתקפה כאשר יש לתוקף גישה גם ל four-tuple וכל שאר המידע מוצפן, נוכל לבצע את הדברים הבאים (בנוסף להצעות שהוצעו מקודם):

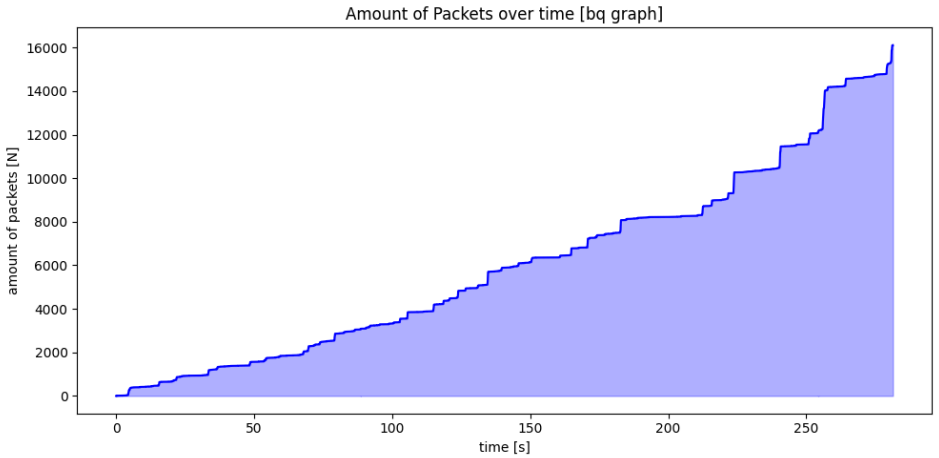
- הלקוח והשרת יכולים להסכים על החלפה של ה ports שבהם הם משתמשים באופן שאינו סדיר ובכך למנוע את הפיכתו של tuple אחד להיות יותר נפוץ מהאחרים.
- הרעש שהלקוח מייצר יכול כעת להיות למספר כתובות IP בכמות יותר קבועה וגדולה ובכך להגדיל את כמות ה tuples המופיעים לעיתים תכופות ולהסוואת יותר טוב את ה tuples האמיתיים

שאלת בונוס:

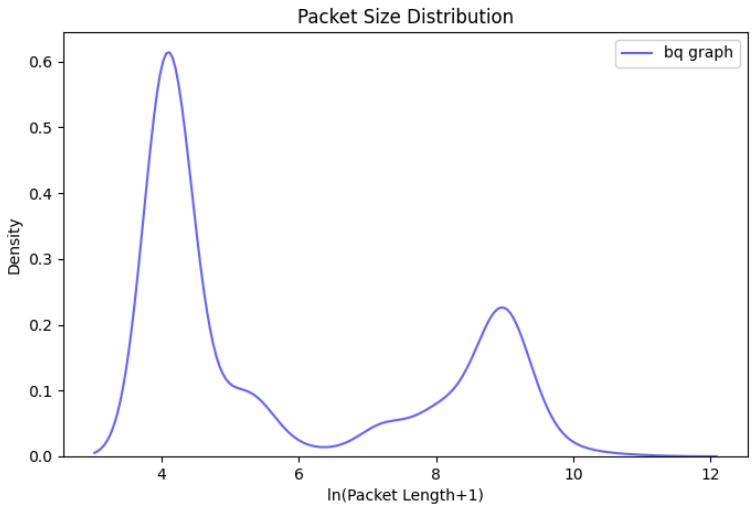
בעבור שאלה זו הקלטנו תעבורה כאשר הפעלנו אפליקציה אחת ראשית ולעיתים השתמשנו בתוכנה משנית, והוצאנו ממנה את המידע הבא:

- גודל החבילות
- ה timestamp שלהן
- Hash של ה Tuple של 4 הערכים הבאים:
  - Source IP
  - Destination IP
  - Source Port
  - Destination Port

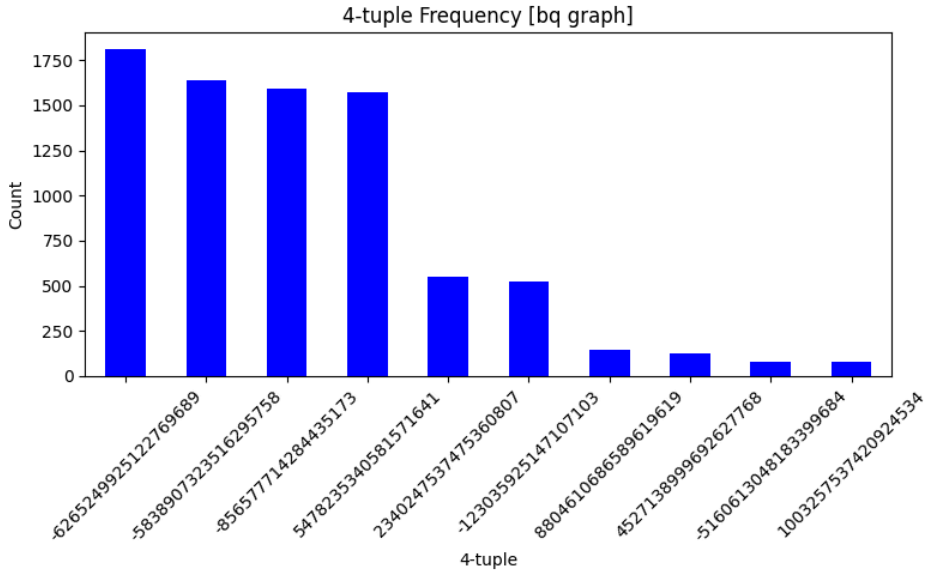
<u>הערה</u>: הקלטנו תעבורה שנעשת על כאשר שומעים מוזיקה ב spotify ועושים חיפושים בויקיפדיה על firefox.



כעת ניתן לראות שכמות החבילות לאורך זמן הינה לינארית יחסית בדומה לגרף של זום אך בקצב גדילה איטי יותר. כלומר יהיה קושי לשייך את הגרף לאחת מהאפליקציות או הדפדפנים הפוטנציאליים.



עם זאת, צפיפות גדלי החבילות לא השתנתה בצורתה משמעותית מאשר הגרפים של chrome, firefox ו spotify ולכן ניתן לצמצם את האפשרויות של התוכנות בהן נעשה שימוש.



בנוסף, מכיוון שישנן מספר אפליקציות בהן השתמשנו באופן סדיר יחסית, לא ניתן לדעת בדיוק מה מה-tuples של התעבורה הראשית והמשנית ולפענח איזה מהם קשורים לתקשורת הראשית ואילו קשורים למישנית, ולכן פיענוח ההבדל ביניהם יצרוך עבודה קשה יותר בלמידת המכונה בשביל להבדיל בין התעבורות השונות ויותר מידע שהאלגוריתם יוכל ללמוד עליו.

זהו בעצם מימוש בסיסי של ההצעות שהצענו בחלק שלוש שאלה 4, המוסיפות "רעש" בתוך התעבורה המסוות בתוכן את התעבורות השונות בצורה המקשה למצוא את ההבדלים ביניהן.