בס"ד

פרויקט גמר - רשתות תקשורת

מגישות:

עדי גם זו לטובה-211602222

שירה מורסיאנו- 322350620

יעל גיל דורני- 212496624

אביטל זר- 331659516

קישור לGITHUB

חלק 1

1. רוחב פס נמוך: במידה ורוחב הפס נמוך, כמות הביטים שעוברת בכל שניה מוגבלת ולכן זמן שהיית הקובץ יעלה. נוכל לבדוק את רוחב הפס ע"י כלים באינטרנט, כמו speedtest.net (יציג את מהירות ההורדה, מהירות ההעלאה וזמן ההשהיה) או לחילופין בדיקה פנימית על ידי ipenf.

זמן השהיה ארוך: במידה והשרת רחוק (מבחינה פיזית) כך שיש מס' נתבים שצריך לעבור או בעיות ב wi-fi כמו תשתית ישנה או הפרעות אלחוטיות ממכשירים נוספים זמן ההשהיה יתארך. ניתן לבדוק זמן השהייה ע"י ping [ן ping IP_dest ן] => time = x ms => זמן ההשהיה במילישניות הוא x.

אובדן חבילות: כאשר אנחנו בפרוטוקול TCP יש מנגנון וידוא הגעת חבילות, במידה וחבילה לא הגיעה יש עיכוב עד הגעת החבילה ולכן זמן ההגעה של שאר החבילות מתארך מה שמשפיע על זמן טעינת הקובץ הכולל. ניתן לבדוק איבוד חבילות ע"י פקודת ping [ן ping -n 50 IP_dest ן].

עומס ברשת: כאשר יש עומס ברשת לנתבים והשרתים קשה להתמודד עם ריבוי החבילות מה שגורם לתור - נוצר תור של חבילות שמחכות לשליחה וככה שהתור ארוך יותר, זמן העברת החבילה יתארך. בנוסף, בפרוטוקול TCP כאשר מורגש עומס יש מנגנון שמוריד באופן יזום את קצב השידור כדי לא להעמיס על השרת. ניתן לבדוק אם יש עומס ברשת ע"י פקודת ping - ן ping -n 100 <IP_dest>l ן

2. מנגנון בקרת הזרימה אחראי לטפל במצבים בהם המחשב השולח הוא בעל יכולת עיבוד גבוהה מיכולת העיבוד של המחשב המקבל.

במצבים אלו, אם לא נשתמש במנגנון בקרת הזרימה, המחשב המקבל יוצף במידע שצריך לעבד שהוא לא מסוגל לטפל בו, מה שיכול לגרום לקריסת מערכת, איבוד חבילות, וחוסר סדר בקבלת החבילות (חבילות יגיעו לא לפי הסדר שבו נשלחו).

המנגנון מייצר "חלון הזזה". החלון מייצג כמות חבילות שיכולות להישלח כל פעם. גודל החלון יקבע על ידי כמות הנתונים שהמחשב המקבל יכול לעבד. כל הודעה תישלח עם מספר מזהה שעל פיו המחשב המקבל ידע לסדר את ההודעות.

המחשב השולח ישלח את המידע לפי כמות הפריטים בחלון ויחכה לקבל ACK מהמחשב המקבל. המחשב המקבל ישלח ACK על כל הודעה שהוא מקבל. הACK יכלול את מספר החבילה שהוא קיבל. אם החבילה מגיעה לא לפי הסדר, הוא ימשיך להחזיר ACK עם המספר המזהה של ההודעה האחרונה שהוא קיבל בסדר, למנגנון זה קוראים "Cumulative ACK".

כאשר המחשב השולח יקבל את הACK הראשון, הוא יקדם את החלון, כך הוא יוכל לשלוח את ההודעה הבאה.

בדרך זו, המחשב השולח ידע אם הודעה לא הגיעה וידע לשלוח מחדש את הנתונים, וגם, המחשב המקבל לא יוצף בהודעות שהוא לא יכול לעבד.

3. על הראוטרים בדרך בה קיימים מספר נתיבים אפשריים, לבחור נתיב עבור כל חבילה שבנוסף להיותו מוביל ליעד, להתחשב במספר גורמים נוספים. בחירה יעילה של נתיב יכולה לפזר את העומס ברשת ולגרום לפחות בעיות תנועה ואיבוד חבילות. בנוסף בחירה מוצלחת תוכל להעביר את החבילה במהירות גדולה יותר, וכך לא רק להוביל לשיפור השירות אלא גם החבילה שנמצאת פחות זמן על הרשת תעמיס עליה פחות וכלל השירות יהיה מהיר ויעיל יותר. בנוסף אם יש תקלה בנתיב, הניתוב ישלח את החבילות לנתיב אחר וימנע אובדן חבילות.

לכן בבחירה של מסלול יש לבחור:

1 . מסלול שיוביל אל היעד, ושאין בו תקלות.

2 . מסלול שצפיפות החבילות בו נמוכה כמה שיותר. כחלק מהגורם הזה, יש לשים לב לרוחב פס: אם רוחב הפס גדול, הצפיפות עשויה להיות נמוכה יחסית לאותו מספר חבילות.

3 . מסלול שאורכו קטן, ושמספר הראוטרים בו, הקפיצות (hops), נמוך.

4. MPTCP הוא מנגנון TCP מורחב. הוא מסוגל לשלוח חבילה אחת דרך נתבים שונים.

בדרך זו, החבילה שלעיתים קרובת מתחלקת לחבילות שונות, תישלח במקביל דרך כמה נתיבים, מה שחוסך את העומס על כל נתב ובנוסף החבילה כולה תגיע מהר יותר כי רוחב הפס של המחשב מנוצל יותר טוב.

שליחת תתי החבילות דרך נתבים שונים נותנת לMPTCP ידע על העומס בנתבים השונים ומאפשר לו לבחור מה הנתיב הפנוי ביותר. במידה ואחד הנתבים קורס, MPTCP יכול להעביר את הנתונים דרך נתבים אחרים, וכך למנוע אובדן חבילות ועיכוב במשלוח.

5. גורמים אפשריים לאובדן חבילות:

עומס רציני - כשיש יותר חבילות משנתב יכול לעבד או להחזיק במטמון בו זמנית. אם מגיעות המון חבילות לנתב שלא יכול לעבד אותן, זה יגרום לאובדן חבילות. אם השולח מסוגל לשלוח כמות גדולה יותר של חבילות משהנתב יכול לקבל, הנתב יאבד חבילות. לחליפין, ייתכן שהנתב מספיק לעבד את כל החבילות אבל רוחב הפס בין הנתבים קטן מכדי להעביר את כולן, וכך חבילות יאבדו בדרך בין שני הנתבים.

בעיות ניתוב - אם יש בעיה בהגדרת הנתיב או שהנתב לא יודע לאן לשלוח את החבילות, החבילות יאבדו. למשל, אם טבלת הניתוב של ראוטר לא מעודכנת לשפצור הרשת האחרון, הוא עלול לקבל כתובת שלא קיימת אצלו, לא ידע מה לעשות עם החבילה ויזרוק אותה.

בעית ניתוב נוספת היא לולאה: אם יהיה חוסר תיאום בין ראוטרים לגבי המסלול הטוב ביותר, עלול להיווצר מצב שהראוטרים שולחים את החבילות זה לזה הלוך וחזור, והחבילות לעולם לא יגיעו ליעדן.

נוסף לאלה, יתכנו בעיות בNAT, וכך כתובת של חבילה תתורגם לא נכון, ותישלח ליעד שגוי או תאבד לחלוטין, אם הIP המתורגם פג תוקף או לא קיים.

<u>פתרונות אפשריים:</u>

אם יש עומס, כי נשלחות יותר חבילות משהנתב מסוגל לעבד בו זמנית, פיתרון אפשרי יהיה להשתמש בפתרונות של פרוטוקול TCP, שמוודא שלא נשלחות יותר חבילות משאפשר לקבל. למשל, באמצעות חלון וACK, שמוודא שהגיעו כל החבילות הראשונות לפני שהוא שולח את הבאות. אפשר להתקין נתיבים חלופיים, כדי שלא תהיה תלות בראוטר קטן.
אם רוחב הפס קטן מדי, שוב נוכל לוודא שלא נשלחות יותר חבילות משאפשר להעביר, או להרחיב את הפס - פיזית או להוסיף נתיבים מקבילים, או כאמור, להוסיף נתיבים חלופיים בשביל למנוע תלות בפס ברוחב קטן מדי.
אם מדובר בשימוש בTCP, נוכל להקטין את גודל החלון כדי להפחית עומס.

אם יש בעית ניתוב - אם טבלת הניתוב לא מעודכנת, צריך פשוט לעדכן אותה. לטווח ארוך, אפשר להשתמש במנגנון כמו RIP שמעדכן את טבלאות הניתוב של ראוטרים אוטומטית.
אם יש לנו לולאת ניתוב, נוכל לנקוט בכמה שיטות, ביניהן:באופן בסיסי יותר, נשתמש בTTL כדי להפחית את העומס והבעיות שחבילות בלולאה יוצרות. למנוע מחבילה לחזור במסלול שהיא באה ממנו (מנגנון split horizon), או לסמן את המסלול שהחבילה עברה בו, ולמנוע מהחבילה לעבור על מסלול מסומן (route poisoning).

חלק 2

מאמר 1: Early_Traffic_Classification_With_Encrypted_ClientHello_A_Multi-Country Study

מהי התרומה העיקרית של המאמר:

פיתוח של אלגוריתם חדש בשם hRFTC (או: hybrid Random Forest Traffic Classifier) שנועד לסיווג תעבורה מוצפנת. סיווג של תעבורה מוצפנת נדרש כדי לייעל את התעבורה ברשת. אנחנו רוצים לדעת איפה יש עומס, אילו חבילות אובדות פעמים רבות, באילו מקומות ברשת אפשר לייעל את התעבורה.

לשם כך, אנחנו נעקוב אחרי חבילות, נפענח את הפרטים הדרושים לנו (זמני RTT, כשלונות בהקמת חיבורים, גודל חבילות ממוצע וכו') וננתח אותם כדי להבין איך לייעל את התעבורה.

האלגוריתם המוצע הוא היברידי, כי הוא משתמש גם בנתונים מחבילות וגם בסטטיסטיקות זרם. הוא חוקר הודעות שהן חלק מתהליך לחיצת ידיים, ומספק סיווג מדויק ויעיל של תעבורה מוצפנת, אפילו בתרחישי Encrypted ClientHello (או: ECH). הוא עובד לא רק על חבילות TLS אלא גם על QUIC ועובד במדינות שונות ברחבי העולם. יכול להתאים את עצמו לרשתות חדשות באמצעות למידה.

באילו מאפייני תעבורה המאמר משתמש, ואילו מהם חדשניים?

מאפיינים קלאסיים שהמאמר משתמש בהם:

נתונים מהודעות ה-TLS הלא מוצפנות, מהודעות client hello.
מאפיינים של זרמים, כמו מעקב אחרי גדלי חבילות ושינויים פתאומיים, או זמני הגעה בין חבילות ומציאת ממוצע, סטיית תקן וכדומה.

מאפיינים חדשניים שהמאמר משתמש בהם:

שילוב של מאפיינים מבוססי חבילות ומבוססי זרמים, שמאפשר דיוק בתנאים מורכבים, למשל מגדיל את האיכות הסיווג של ECH.
מאפיינים חדשניים של זרמים: מספר חבילות - מאפיין שלא משתמשים בו בד"כ, סיווג חבילות לפי היסטוגרמה.
מנגנון בחירת חבילות מתקדם, שבוחר חבילות טוב ככה שאיכות הסיווג תעלה בלי לפגוע בזמן איסוף המידע.
כשבוחנים חבילות, אחד החסרונות זה האיזון שצריך בין דיוק - בשביל דיוק גבוה יש צורך בחבילות רבות, לעיכוב תעבורה - בחינת חבילות רבות מעכבת את התעבורה. ההצעה של האלגוריתם החדש היא לנתח את כל החבילות של הhandshake, שיש בהן הרבה מידע יחסית והן לא רבות מספיק כדי לעכב מדי.

תוצאות:

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

בטבלה הזו ניתן לראות את אחוזי הדיוק של מספר אלגוריתמים בתעבורה לאתרים שונים. ניתן לראות כמובן, שהאלגוריתם החדש המוצע עובד טוב יותר בכל הזדמנות, ובאופן ספציפי, הוא עובד מצוין גם במקומות בהם רוב האלגוריתמים נכשלים ברמת דיוק גבוהה. למשל, ב SBV instagram שני האלגוריתמים ההיברידיים המצוינים מצליחים רק ברמת דיוק של כשבעים וחמישה אחוזים, בעוד האלגוריתם החדש מצליח לדייק בכמעט 90 אחוזים. כך ניתן לראות שאלגוריתם מותאם טוב יותר לשוני בין סוגי תעבורה.

נוסף לאלה, תובנה חשובה מהמאמר היא שאלגוריתמים היברידיים משפרים מאוד את הסיווג. זה בעצם הרעיון של האלגוריתם: לקחת את האלגוריתם RB-RF המבוסס-פקטות, ולשכלל אותו שיהיה מבוסס גם על זרימה. ואכן, האלגוריתם החדש יעיל יותר מפי שניים בממוצע מ-RB-RF.

מאמר 2:

FlowPic_Encrypted_Internet_Traffic_Classification_is_as_Easy_as_Image

Recognition

מהי התרומה העיקרית של המאמר?

המאמר מציע דרך פשוטה אך גאונית לסיווג תעבורה לקטגוריות, כשבמקום להתעמק בפרטי החבילות או בנתוני זרימת רשת רבים, ניתוחם והגעה לתוצאות מדויקות ככל האפשר, היא משתמשת בשני פרמטרים: גודל החבילה וזמן ההגעה שלה. את הנתונים המתקבלים מציירים על גרף, ואז הניתוח נעשה באמצעות CNN, כלי שאנחנו משתמשים בו הרבה ומפתחים כל הזמן: מנתח תמונות באמצעות בינה מלאכותית. במקום לפתח כלים יעודיים לתעבורת רשת, יש כאן שילוב כוחות והתאמת הבעיה לבעיה של מישהו אחר, וכך ניתן לפתור את שתיהן יחד. ההתקדמות של השיטה הזו תהיה בהתאם להתקדמות המודלים לניתוח תמונות, שמתפתחים במהירות כל הזמן. CNN לוקח את התמונה ומוצא דפוסים, בין חלקים שונים בה ובינה לבין תמונות אחרות, ומצליח לגלות מה סוג התעבורה: העברת קבצים, חיפוש ביוטיוב וכו'.
"תיקח את הקומקום למטבח ושם כבר פתרנו".

באילו מאפייני תנועה משתמשים, ואילו מהם חדשניים?

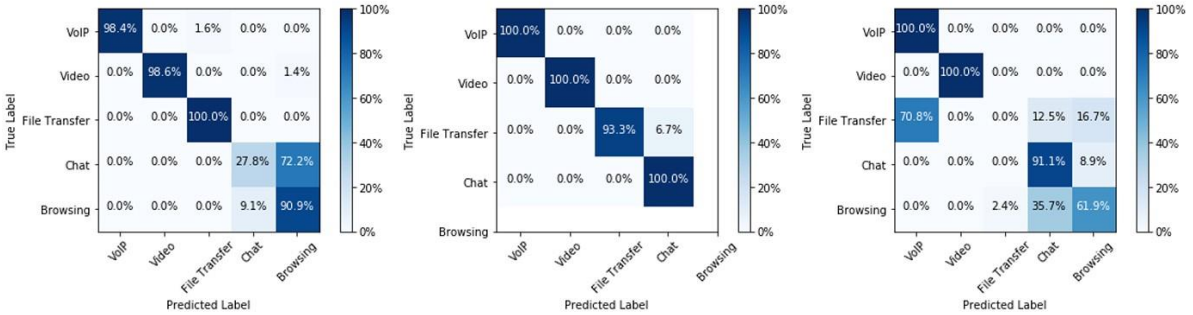
השיטה המוצגת במאמר משתמשת בשני מאפייני תנועה: גודל החבילות וזמן ההגעה שלהן. אף אחד מהם לא חדשני, החדשנות היא מה שעושים איתם אחר כך. זו חלק מהגאונות בשיטה. היא לא משתמשת בנתונים לא רגילים או פורצי דרך, אלא עושה שימוש יעיל בהרבה בנתונים הבסיסיים ביותר.

מהן התוצאות העיקריות, ומה התובנות מהתוצאות שלהם?

Problem	**FlowPic Acc. (%)**	**Best Previous Result**	**Remark**
Non-VPN Traffic Categorization	85.0	84.0 % Pr., Gil *et al.* [15]	Different categories. [15] used unbalanced dataset
VPN Traffic Categorization	98.4	98.6 % Acc., Wang *et al.* [7]	[7] Classify raw packets data. Not including browsing category
Tor Traffic Categorization	67.8	84.3 % Pr., Gil *et al.* [15]	Different categories. [15] used unbalanced dataset
Non-VPN Class vs. All	97.0 (Average)	No previous results	
VPN Class vs. All	99.7 (Average)	No previous results	
Tor Class vs. All	85.7 (Average)	No previous results	
Encryption Techniques	88.4	99. % Acc., Wang *et al.* [7]	[7] Classify raw packets data, not including Tor category
Applications Identification	99.7	93.9 % Acc., Yamansavascilar *et al.* [10]	Different classes

בטבלה נבדקת רמת הדיוק של הסיווג לקטגוריות לכל סוג תעבורה - לא מוצפנת, עם VPN וTOR, איך האלגוריתם הזה הצליח בסיווג לעומת האלגוריתמים הקודמים. קודם כל, בסיווג רגיל, הדיוק מגיע לרמת הדיוק הקודמת, חוץ מעם TOR שהוא מוצפן יותר. כשבודקים את רמת הדיוק, כשאומרים לאלגוריתם לבדוק "האם התעבורה היא מסוג כך וכך או לא" (class vs. all), הוא עובד אפילו טוב עוד יותר.

החלקים האחרונים: סיווג טכנולוגית הצפנה - פחות טוב מממצאים קודמים.
אבל אחרי שסיווגנו לפי קטגוריה, הסיווג לפי יישום טוב בהרבה מממצאים קודמים ונותן סיווג קרוב למושלם.



(a) Non-VPN (b) VPN (c) Tor

בגרף זה, באופן ספציפי יותר, ניתן לראות את התוצאות שסוכמו בטבלה קודם. Tor כאמור הוא מסווג יותר, אבל כאן ניתן גם לראות שהטעויות שהתרחשו חזרו על עצמן בדברים ספציפיים. כלומר, נניח בNon-VPN האלגוריתם טעה שוב ושוב בזיהוי chat. וב Tor, האלגוריתם תמיד פירש את העברת הקבצים לא נכון, בדרך כלל כ-VoIP.

נראה את רמת הדיוק של האלגוריתם להבחנה בין יישומים שונים כאשר ידועה הקטגוריה - video או volP.

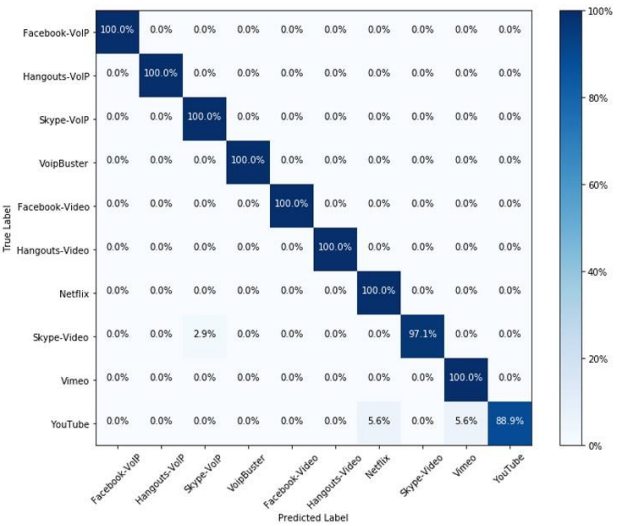


Figure 5: A confusion matrix of the VoIP and video applications identification problem.

רמת הדיוק היא גבוהה מאוד מאוד, כשאלגוריתמים קודמים השיגו רק כ-90% הצלחה במקרים כאלה. כמעט בכל המקרים, האפליקציות שניסו, האלגוריתם הצליח לחלוטין, מלבד skype שם ההצלחה היא כמעט מוחלטת ו youtube שם יש כעשירית של טעויות.

בסך הכל, ניתן לראות שהשיטה הזו עובדת באופן מרשים ביותר ומצליחה לפענח תעבורה לעיתים כמו השיטות הקודמות (רק עם הרבה פחות נתונים!), לעיתים פחות ולעיתים קרובות אפילו יותר. אפשר להסיק ש flowPic עובדת הכי טוב על סיווג ליישומים ספציפיים, ובאופן טבעי היא נכשלת יותר כשמדובר בתעבורה מוצפנת יותר.

ואם נזכור שמדובר באלגוריתם שמשתמש בנתונים בסיסיים ביותר ולא פוגע באבטחה של אף חבילה, לעומת האלגוריתמים הקודמים שהיו צריכים להשתמש בהמוני סטטיסטיקות ולהשתמש בכל המידע שכל חבילה יכלה לתת, ונוסף על כך, האלגוריתם עומד להתקדם יותר ויותר ככל שתחום הבינה המלאכותית יתפתח, אין ספק שיש כאן תוצאות מרשימות ביותר.

מאמר מספר 3:

,Analyzing HTTPS Encrypted Traffic to Identify User's Operating System

Browser and Application

1.התרומה המרכזית של המאמר היא הוכחת היכולת לזהות את מערכת ההפעלה, הדפדפן והיישום של משתמשים מתוך תעבורת HTTPS מוצפנת, באמצעות למידת מכונה וניתוח מאפייני תעבורה ייחודיים.

ההוכחה מתבססת על שימוש במאפיינים חדשים של תעבורה, כולל דפוסי התנהגות פורצת (bursty behavior) של דפדפנים ומאפייני SSL/TLS אשר נבדקים על מערך נתונים מקיף של למעלה מ-000,20 דוגמאות מתויגות. תוצאות המחקר מציגות דיוק גבוה של 06.96% בזיהוי, הישג שלא הוצג במחקרים קודמים.

<u>2 מאפייני תעבורה סטנדרטיים:</u>

1 . מספר חבילות בזרימה קדימה ואחורה (Forward/Backward Packets).

2 . כמות הנתונים המועברת לכל כיוון.

. פרקי זמן בין חבילות (Inter-Arrival Time) . 3

3 . סטטיסטיקות גודל חבילות (מינימום, מקסימום, ממוצע, סטיית תקן).

<u>מאפייני תעבורה חדשים :</u>

1 . מאפייני SSL/TLS:

- מספר שיטות הצפנה (Cipher Methods) הנתמכות ב- SSL.
- מספר ההרחבות (Extensions) בפרוטוקול SSL.
- גודל מזהה הסשן של (SSL (SSL Session ID Length .

2 . התנהגות פורצת של דפדפנים (Bursty Behavior):

- כמות הפיקים (bursts) בתעבורה.
- זמן בין שיאים (Peaks).
- מקסימום ומינימום קצב העברת הנתונים.

3.ממצאים מרכזיים:

- המאמר מוכיח כי ניתן לזהות מערכת הפעלה, דפדפן ויישום מתוך תעבורה מוצפנת בדיוק גבוה.
- המאפיינים החדשים משפרים משמעותית את ביצועי המערכת, כאשר שילוב של מאפיינים בסיסיים + חדשים נותן את הביצועים הטובים ביותר.
- מערכת ההפעלה זוהתה כמעט ללא טעויות, בעוד שזיהוי היישום היה מאתגר יותר, במיוחד עבור פייסבוק.

תובנות מהתוצאות:

- גם עם הצפנה (HTTPS) , ניתן לחלץ מידע משמעותי על המשתמש מתעבורת הרשת.
- תוקפים יכולים לנצל שיטה זו כדי להתאים מתקפות ספציפיות למשתמש לפי המערכת והיישומים שבהם הוא משתמש.
- הצפנה אינה מספיקה לשמירה על פרטיות המשתמשים, ויש צורך בטכניקות נוספות להגנה מפני ניתוח תעבורה.

בס"ד

חלק 3 - סיווג אפליקציות על פי מאפייני תעבורה

תהליך הקלטת התעבורה דרך הWireshark:

אפליקציית Youtube

התחברנו לYouTube דרך chrome והתחלנו את ההקלטה,עברנו בין שני שירים אחד עם קליפ ואחד בלי והרצנו קדימה ואחורה את הסרטונים

דפדפן Chrome

התחברנו לדפדפן chrome וגלשנו לאתרים שונים כגון Facebook,Wikipedia,google וכו׳ (בדגש על בחירה באתרים שלא נבדקים כמו zoom)

דפדפן FireFox

התחברנו לדפדפן FireFox וגלשנו לאתרים שונים כגון Instagram,Wikipedia וכו׳ (בדגש על בחירה באתרים שלא נבדקים כמו zoom)

אפליקציית Zoom

יצרנו דרך המשתמש בwindows שיחת zoom הכוללת וידאו ואודיו חי ושלחנו קישור התחברות למשתמש בUbuntu אשר שידר אודיו וקלט וידאו ואודיו מהמשתמש השני

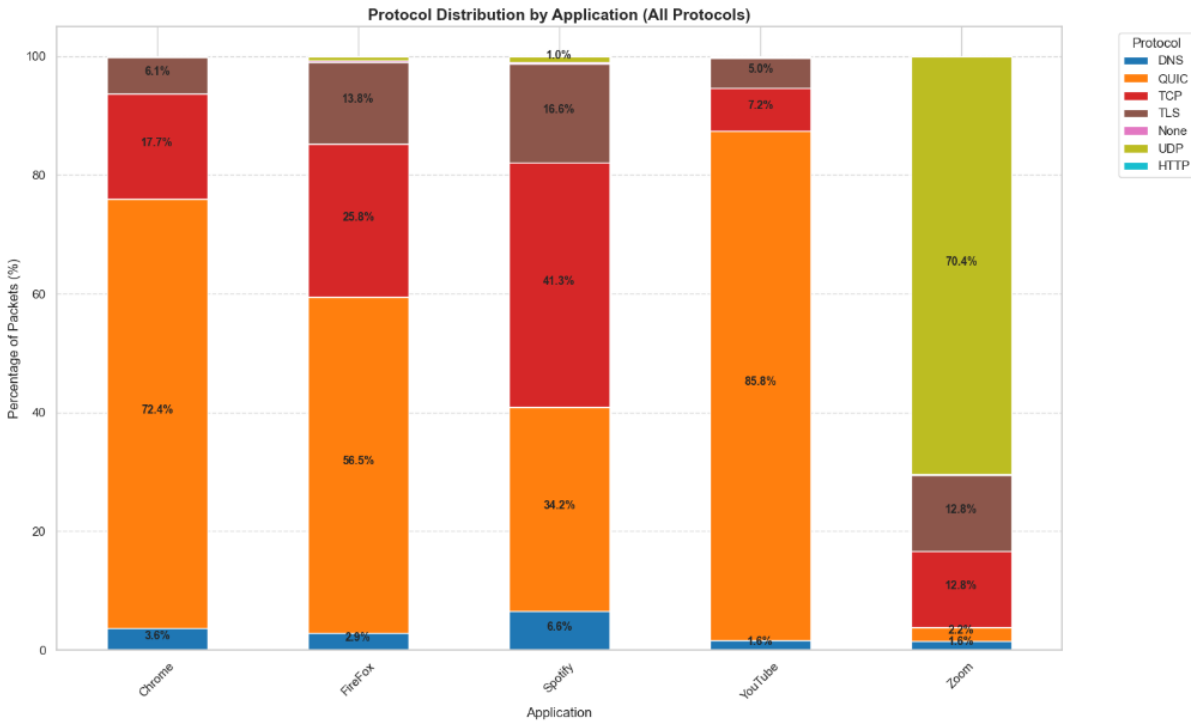
יישום Spotify

התחברנו למשתמש בSpotify וניגנו שני חצאי שירים שונים,עברנו בין מסך תמונה למסך המילים של השיר והרצנו קדימה ואחורה את השירים

בס"ד

חלק 3 ניתוח הגרפים:

PROTOCOLS GRAPH



הגרף מציג את התפלגות הפרוטוקולים בכל אפליקציה. ניתן לראות כי TCP ו-QUIC הם הדומיננטיים ברוב האפליקציות, למעט Zoom, שבו UDP הוא הפרוטוקול המרכזי (70.4%) בשל הצורך בהעברת מידע במהירות וללא עיכובים.

Chrome, Firefox, Spotify, ו-YouTube מסתמכות בעיקר על TCP ו-QUIC, כיוון שהן פועלות בעיקר בדפדפן, שם נדרשת תקשורת אמינה ומסודרת.

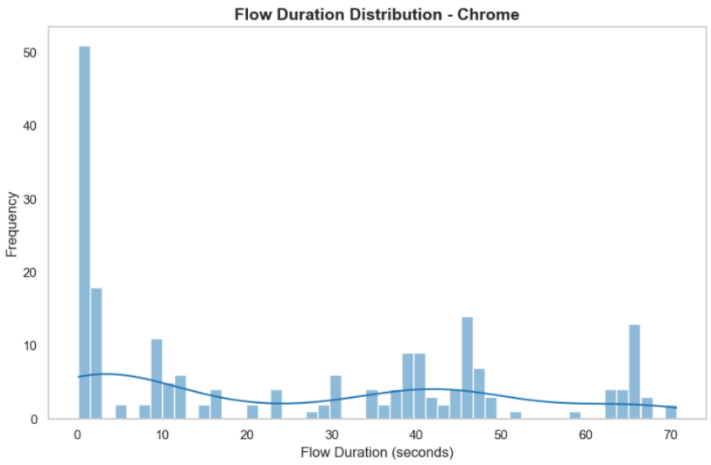
QUIC בולט במיוחד ב-Chrome וב-YouTube, שכן מדובר בפרוטוקול מודרני של גוגל המשפר ביצועים, במיוחד בהזרמת וידאו.

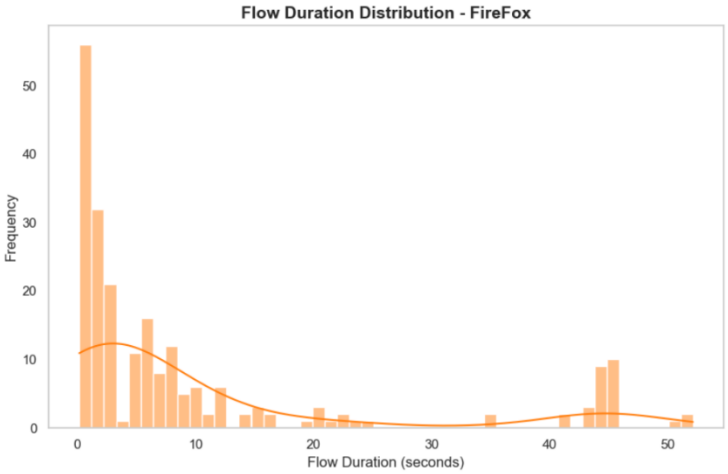
Zoom משתמש בעיקר ב-UDP (ולא ב-TCP), מכיוון שפרוטוקול זה מקטין השהיות ומאפשר שיחות וידאו ושמע בזמן אמת.

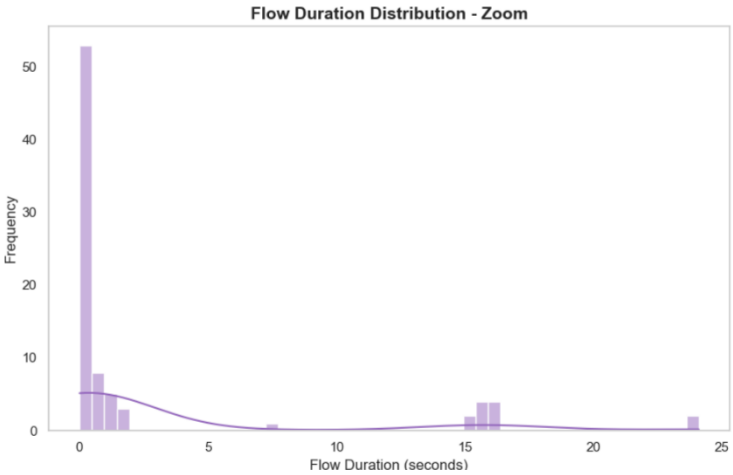
ההבדלים בפרוטוקולים משקפים את הצרכים הייחודיים של כל אפליקציה – אמינות מול מהירות תגובה.

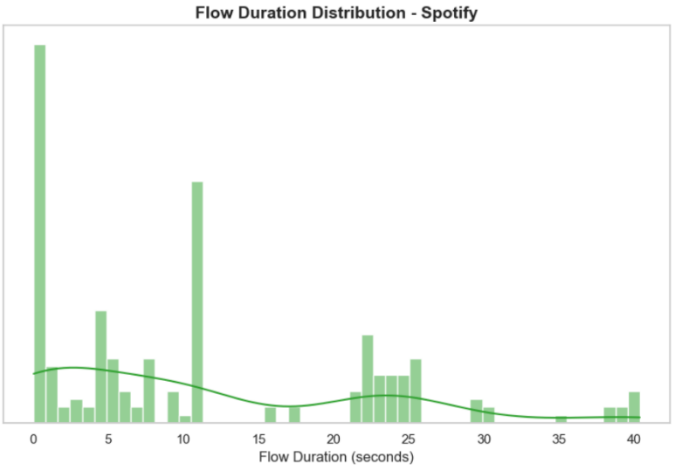
בס"ד

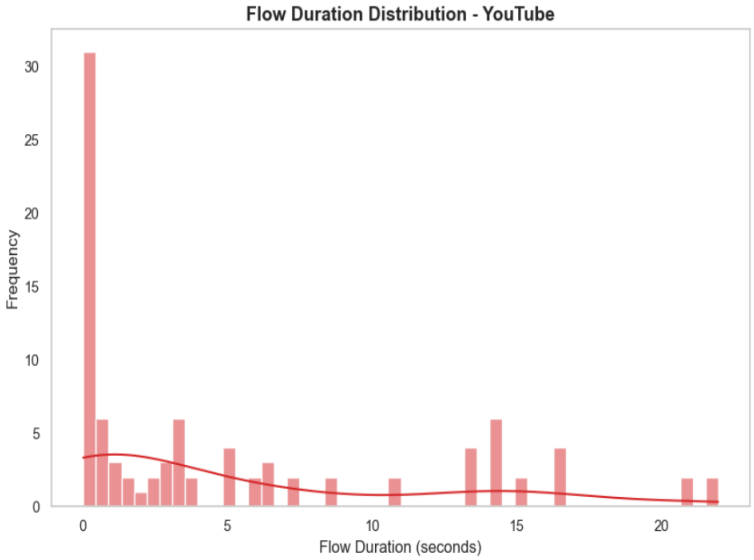
Flow Lengths Graphs











זרימה (Flow) מוגדרת כרצף של חבילות ברשת החולקות 5 מאפיינים ייחודיים (Tuple-5):

1 **כתובת Source IP** – כתובת השולח

2 **כתובת Destination IP** – כתובת המקבל

3 **פורט מקור** – מספר הפורט של השולח

4 **פורט יעד** – מספר הפורט של המקבל

5 **פרוטוקול** – למשל TCP, UDP

<u>זמני ניתוק (Timeouts)</u>

- Active Timeout: אם חיבור נמשך זמן רב, הוא מחולק למקטעים (למשל, כל 5 דקות).
- Inactive Timeout: אם אין חבילות חדשות למשך זמן מסוים (למשל, 30 שניות), הזרימה נסגרת.

<u>כיצד מזהים סיום זרימה?</u>

- ב-TCP: החיבור נסגר באופן יזום ע"י FIN/ACK או RST
- ב-UDP: הזרימה נסגרת כאשר אין תעבורה לזמן מסוים (Timeout)

השוואת אופן הזרימה ב-YouTube וב-Spotify

- YouTube מבצע buffering, שבו קטעי וידאו נטענים מראש כדי למנוע השהיות.
- Spotify משתמש בהזרמה רציפה במקום טעינה מוקדמת, כיוון שהעברת אודיו דורשת רוחב פס קטן יותר מווידאו.
  כתוצאה מכך:

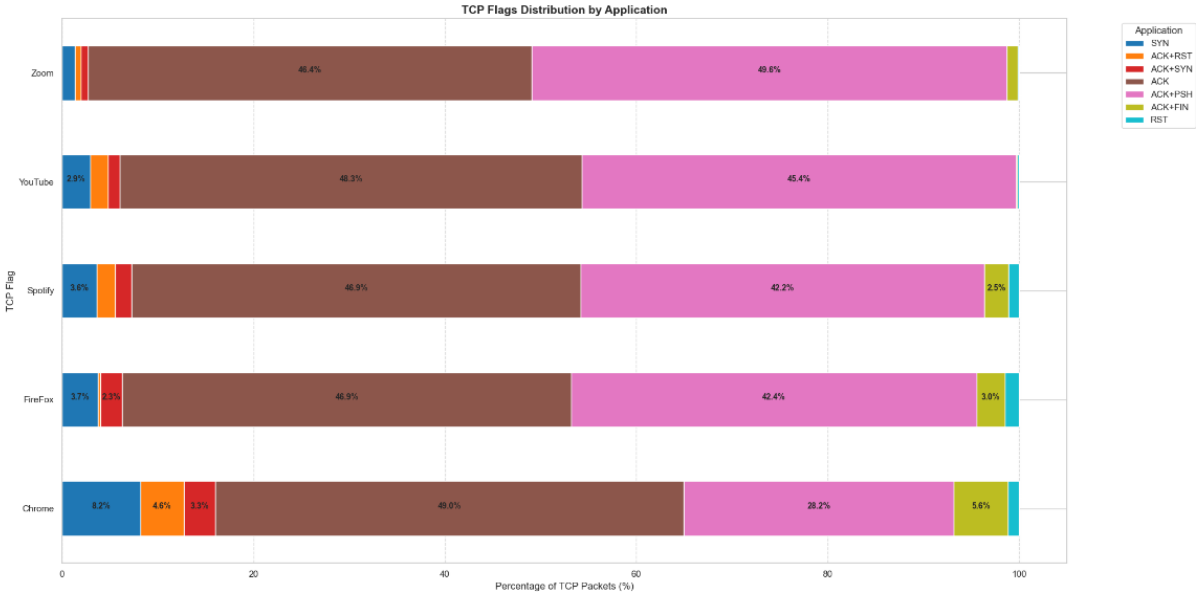
ביוטיוב – רוב החיבורים קצרים ולא רציפים.

בספוטיפיי – החיבורים ארוכים ומתרחשים בקפיצות חדות.

השוואה מול שאר האפליקציות

רוב היישומים דומים ליוטיוב מבחינת מבנה החיבורים: חיבורים קצרים ולא רציפים.
Zoom וגלישה בדפדפנים כמו CHROME ו FIREFOX נשענים על חבילות מידע קטנות (bite-size data), ולא על הזרמת מדיה כבדה. לכן, משך הזרימה קצר יותר משמעותית בהשוואה ל-YouTube ו-Spotify.

TCP FLAGS GRAPH



הגרף מציג את ההתפלגות של דגלי TCP.
כל צבע מייצג דגל שונה, ניתן לראות כי Zoom, YouTube ו-Spotify מציגים דפוסים דומים, עם רוב החבילות הנושאות את הדגלים ACK ו-ACK+PSH, דבר שמצביע על תקשורת זורמת של נתונים

ב-Zoom יש מעט חבילות SYN ורובן המכריע מסווגות כ-ACK+PSH, מה שמעיד על זרימה מתמשכת של נתונים ללא יצירת חיבורים חדשים רבים. ב-YouTube ההתפלגות דומה אך עם מעט יותר חבילות SYN, מה שמעיד על הקמת חיבורים חדשה לעיתים.

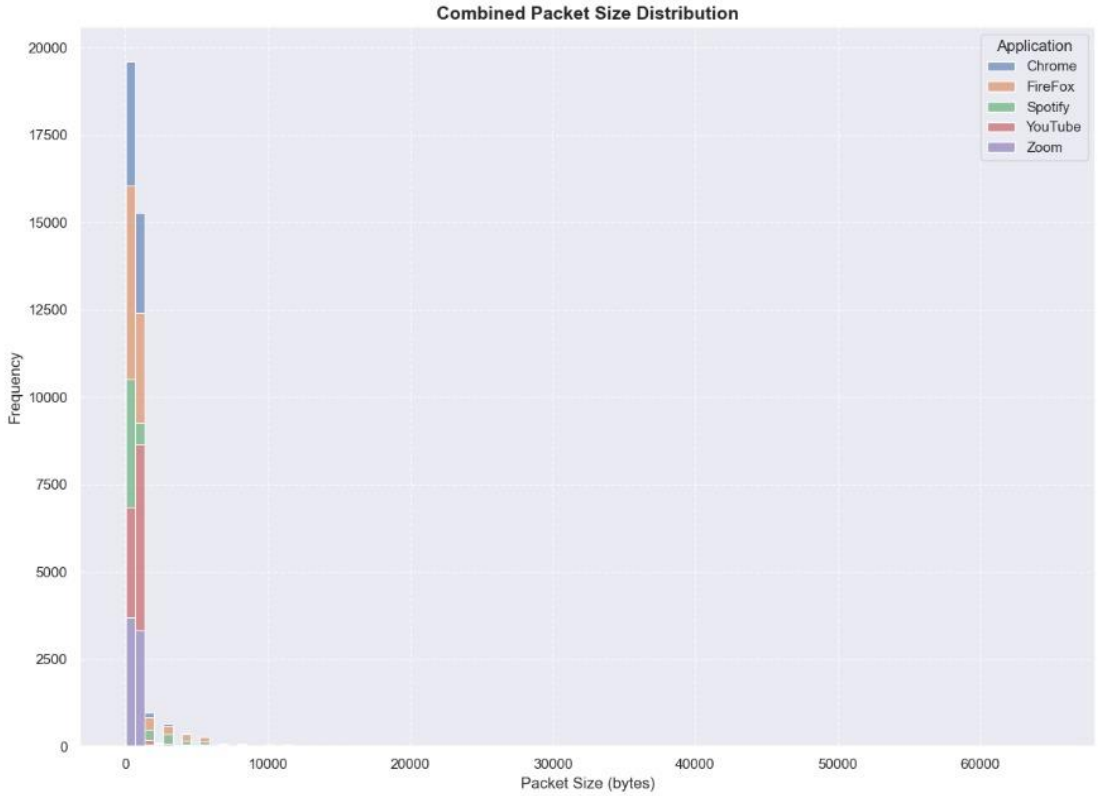
Spotify דומה מאוד במבנה, אך מכיל פחות ACK+PSH בהשוואה לYouTube.

FireFox ו-Chrome, לעומת זאת, מציגים יותר חבילות SYN ו-RST, מה שמעיד על פתיחה וסגירה תכופה של חיבורים, דבר שמאפיין גלישה באינטרנט שבה נטענים דפים רבים בזמן קצר. ב-Chrome ניתן לראות אחוז גבוה יחסית של חבילות FIN ו-RST, דבר המעיד על ניתוקי חיבורים

תכופים וסיום חיבורים יזום על ידי הדפדפן. Firefox דומה בהתנהגותו אך מציג מעט פחות חבילות RST בהשוואה ל-Chrome.

הגרף מדגים כיצד ניתן לזהות שימוש ביישומים שונים לפי דפוסי דגלי ה-TCP שלהם, כאשר סטרימינג ושיחות וידאו מאופיינים בזרימה קבועה של חבילות ACK+PSH, ואילו דפדפנים מציגים יותר חבילות SYN, FIN ו-RST.

PACKET SIZE GRAPH



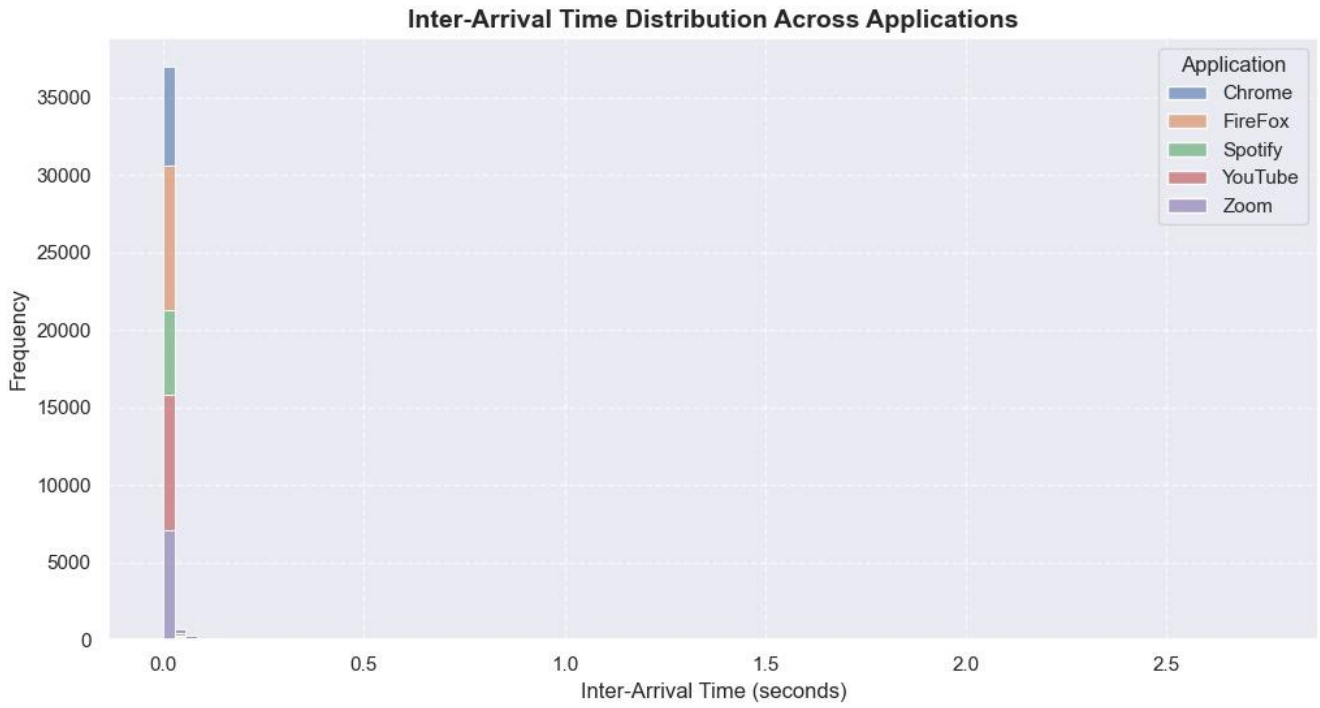
הגרף מציג את התפלגות גודל חבילות הנתונים ביחס לתדירות שליחתן.

הציר האופקי מייצג את גודל החבילה (בבתים), ואילו הציר האנכי מייצג את מספר המופעים של כל חבילה בגודל מסוים.

ניתן לראות כי רוב החבילות קטנות יחסית- מרבית התדירות מרוכזת בערכים נמוכים של גודל חבילות (פחות מ-10,000 בתים), וההתפלגות יורדת במהירות ככל שגודל החבילה גדל.

נתון זה מעיד כי רוב התקשורת בין האפליקציות מבוססת על מנות קטנות יחסית אך בתדירות גבוהה,באפליקציות כמו chrome,FireFox דבר זה מתקיים עקב שינוי בקצב גבוה של נתונים(פתיחת דפדפן חדש,גלילה למטה בעמוד וכו׳) לעומתם zoom,YouTube וSpotify ישמרו על שידור של חבילות קטנות בתדירות גבוהה כדי לשמור על רצף אחיד בהפעלת וידאו/אודיו.

INTER ARRIVAL TIME GRAPHE



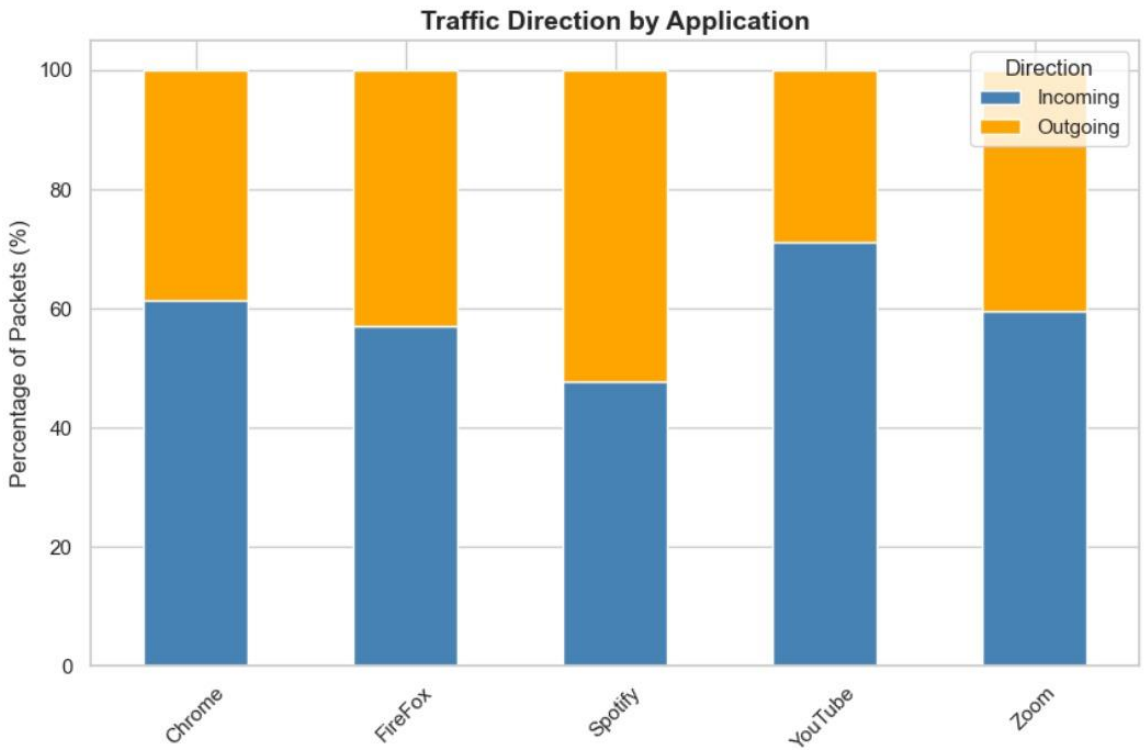
הגרף מציג את התפלגות זמני ההגעה בין חבילות נתונים ביחס לתדירות הגעתן.

הציר האופקי מייצג את זמן ההגעה בין החבילות בשניות– כמה זמן עבר בין חבילה אחת לבאה והציר האנכי מייצג את מספר הפעמים שכל ערך של זמן הגעה הופיע בנתונים.

ניתן לראות כי רוב זמני ההגעה קצרים מאוד כאשר ריכוז הנתונים גבוה מאוד סביב ערך 0 שניות,מעט מאוד חבילות עם זמני הגעה ארוכים – ככל שזמן ההגעה בין החבילות גדל, התדירות יורדת באופן דרמטי, כמעט לא קיימות חבילות עם הפרשי זמן העולים על 0.5 שניות.

ההתנהגות הזו אופיינית ליישומים בעלי תקשורת רציפה, כמו שירותי סטרימינג (YouTube, Spotify), שיחות וידאו (Zoom), וגלישה בדפדפנים (Chrome, Firefox).

למשל, Zoom נדרש להעביר מידע כל הזמן ולכן יש מנות שמגיעות בקצב גבוה מאוד.YouTube ו-Spotify מעבירים נתוני מדיה באופן רציף אך עשויים לכלול הפסקות מסוימות ודפדפנים (Chrome, Firefox) מעבירים מנות בהתאם לפעילות המשתמש, מה שיכול לגרום להתפלגות מגוונת יותר.



בchrome ובFireFox כמות התעבורה הנכנסת גדולה מהתעבורה היוצאת, מפני שהדפדפן בעיקר מוריד מידע למחשב ובד"כ המידע יחסית גדול וגורם לכניסת נתונים רבה אל המחשב. החבילות היוצאות הן בקשות לשרתים אחרים והעלאות נתונים (שקורות פחות מהורדת נתונים..).

גם בYoutube יש יותר חבילות נכנסות.

נשלחת בקשה לסרטון מסוים, מהמחשב לשרת (חבילה יוצאת) ובתמורה השרת מחזיר למחשב את הסרטון מחולק לחבילות (הסרטון בד"כ גדול מדי בשביל להישלח כחבילה 1). לכן, יש יותר חבילות נכנסות מיוצאות.

בSpotify, יש איזון בין כמות החבילות היוצאות לנכנסות, מפני שאודיו יותר קטן מסרטון ולכן צריך פחות חבילות לכל בקשה.

בZoom היינו מצפים שיהיה איזון בין כמות החבילות הנכנסת ליוצאת, כי התעבורה היא דו כיוונית, אבל לפי הגרף, שוב יש יותר הודעות נכנסות. יכול להיות שאולי המיקרופון/ של המחשב שבו הוקלטה התעבורה היה כבוי, אז לא נשלחו חבילות אודיו/ סרטונים, בנוסף, יכול להיות שהייתה בעיית קליטה במחשב שלא שלחה את כל הנתונים.

סעיף 4 - התוקף.

אפשרות ראשונה: התוקף יודע את גדלי החבילות, את חותמת הזמן ונוסף על כך, את ה flow id (כתובות מקור ויעד ופורטים).

כיוון שהתוקף יודע את כתובת ה IP של מקור ויעד החבילה, הוא יוכל לדעת בדיוק איזה משתמש משתמש באילו יישומים. הוא יוכל לחפש למי שייכת כתובת היעד ולבדוק על איזה יישום מדובר. למשל, אפשר להשתמש ב netstat וכלים נוספים של שורת הפקודה שיכולים לאתר אתר על פי כתובתו.
במקרים מסוימים, ייתכן שידיעת כתובת הIP לא תספיק. למשל, ישנם יישומים שמשתמשים כמה יישומים באותה כתובת IP, ואיתורה לא ימצא את היישום המדויק שמשתמש בו. או לחליפין ייתכן שמדובר בכתובת IP דינאמית, שבעליה משתמש בה אך לזמן קצר, ושוב איתורה לא ישיג את היישום האמיתי שהמשתמש ניגש אליו. ואז, התוקף ישתמש רק בגודל החבילה וחותמת הזמן, בדיוק כמו התוקף מהאפשרות השנייה, שנראה מיד.

אפשרות שנייה: התוקף יודע רק את גדלי החבילות ואת חותמת הזמן.

במידה והמידע היחיד הקיים בידי הוא גודל החבילה וחתימת הזמן שלה נוכל להשוות בין מאפייני ההודעות השונות המגיעות כך:

גודל החבילה:

יישומים כמו youtube ישלחו חבילות גדולות מכיוון שyoutube הוא שירות סטרימינג משמע הרבה מידע שאמור לעבור במהירות עם איכות גבוהה, כדי לא לבזבז רוחב פס עדיף לyoutube לשלוח חבילות גדולות וכבדות וככה בנוסף לניצול רוחב פס, החבילות הגדולות מאפשרות טעינה מהירה ככה שזמן השהייה יקטן בעקבות זה.

Spotify ישלח חבילות קטנות עד בינוניות, כיוון שקבצי שמע דורשים פחות נתונים בהשוואה לווידאו. גודל החבילה ישתנה בהתאם לאיכות המוזיקה ולרוחב הפס הזמין, אך באופן כללי הן יהיו קטנות יותר מאלו של שירותי סטרימינג וידאו כמו של youtube.

Zoom לעומתם ישלח חבילות קטנות יחסית מכיוון שמדובר בשיחות וידאו ואודיו בזמן אמת. כדי להימנע מעיכובים ולשמור על איכות שיחה חלקה, האפליקציה שולחת הרבה חבילות קטנות במקום חבילות גדולות וכבדות. כך ניתן לצמצם את ההשהיה ולוודא שהמידע מתקבל במהירות האפשרית ובאיכות טובה.

Chrome בשונה מהקודמות ששלחו באופן אחיד ישלח דווקא חבילות בגודל משתנה מאוד, כיוון שהוא דפדפן שמבצע טעינה של משאבי רשת שונים כגון קוד HTML, תמונות, סקריפטים וקובצי CSS. בעת טעינת אתר

חדש תישלחנה חבילות גדולות יותר, אך לאחר מכן ייתכנו גם חבילות קטנות של עדכונים בעמוד ותקשורת רקע עם השרת.

Firefox בדומה לchrome ישלח חבילות בגודל דינמי בהתאם לסוג האתר שבו המשתמש מבקר. הדמיון מתבטא בכך שהוא טוען קבצים בגדלים שונים בהתאם לצרכים של האתר. ייתכנו חבילות גדולות יותר בעת טעינת עמודים כבדים, אך עם תוספי חסימת פרסומות או מעקבים, כמות החבילות עשויה להיות קטנה יותר בהשוואה ל-chrome (מה שיראה לנו את השוני בינהם)

לסיכום, ננסה למקם את גודל ההודעות על ציר בין הודעות גדולות וכבדות עד הודעות מאוד קטנות כדי לקבל אינדיקציה ראשונית האם תוכן ההודעה יכיל הודעות כבדות כמו וידאו או קטנות כמו פרסומות לאתר, בנוסף יצטרף לגודל ההודעה חתימות הזמן שתכף נסביר שיתנו כיוון נוסף של זמן בהתאם לגודל.

חתימת הזמן:

יישומים כמו youtube ישלחו בהתחלה פרץ הודעות מאוד גבוה (עבור טעינה ראשונית של הסרטון) ולאחר מכן הזרמה קבועה של חבילות גדולות כל זמן שהסרטון מתנגן, במידה והמשתמש עצר את הסרטון ההודעות יעצרו אך ברגע שהוא יפעיל חזרה נמשיך לקבל את החבילות הגדולות בקצב קבוע. מבחינת חתימת הזמן אנחנו נראה בהתחלה פרק זמן קצר בין הודעות ואז שינוי לחבילות במרווחי זמן אחידים, תיתכן עצירה כמעט מוחלטת בהודעות(במידה והמשתמש עצר את הסרטון) או פרץ הודעות נוסף כמו בהתחלה (במידה והמשתמש דילג להמשך הסרטון).

Spotify דומה בחלקו לyoutube אך בשונה ממנו הוא ישלח חבילות בקצב מאוד קבוע כל עוד השיר מתנגן, כיוון שהשידור מתבצע בצורה יציבה וללא צורך בהשהיות ארוכות. בתחילת ההשמעה עשוי להיות פרץ ראשוני של חבילות לצורך טעינה מוקדמת של השיר, ולאחר מכן קצב אחיד של חבילות קטנות לאורך זמן. במעבר לשיר חדש תיתכן עלייה רגעית במספר החבילות לפני חזרה לקצב קבוע.

Zoom ישלח חבילות בקצב קבוע ומהיר מאוד לאורך כל זמן השיחה, כיוון שהוא צריך לשמור על זרימה רציפה של וידאו ואודיו. לא יהיו הפסקות משמעותיות בין שליחת החבילות, מכיוון שכל עיכוב מורגש מיד בשיחה. אם אין דיבור או תנועה, ייתכן שקצב החבילות יפחת, אך הן עדיין ימשיכו להישלח כדי לשמור על החיבור פעיל.

Chrome ישלח חבילות בתבנית לא קבועה, כיוון שגלישה באינטרנט אינה אחידה. בתחילת טעינת עמוד תהיה תנועה גבוהה מאוד של חבילות בפרקי זמן קצרים, ולאחר שהעמוד נטען התעבורה תקטן משמעותית. אם המשתמש יגלול למטה או ילחץ על קישור חדש, תתבצע שליחה נוספת של חבילות בבת אחת, אך לא בקצב קבוע כמו בשירותי סטרימינג.

Firefox ישלח חבילות בצורה לא סדירה, עם פרץ ראשוני גדול בעת טעינת דף חדש ולאחר מכן תנועה

אקראית בהתאם לפעולות המשתמש. כאשר אין פעילות גלישה, ייתכנו בקשות רקע אך במספר נמוך יותר מאשר בדפדפנים ללא חסימת מעקבים. אם המשתמש גולל או מקליק, יופיעו קפיצות חדות בקצב שליחת החבילות, אך לא בתבנית אחידה כפי שניתן לראות בסטרימינג או באפליקציות תקשורת.

לסיכום, ננסה למקם את זמן הגעת ההודעות על ציר בין הודעות מתפרצות להודעות בזמן אחיד כדי לקבל אינדיקציה ראשונית האם תוכן ההודעה יכיל הודעות מתפרצות כמו וידאו/דפדפן או הודעות בזמן אחיד כמו אודיו, בנוסף יצטרף גודל ההודעה אשר הוסבר לגביו מקודם כך שבסיכום כולל נקבל:

- **Youtube** פרץ חבילות כבדות בהתחלה ולאחר מכן הזרמה קבועה של חבילות גדולות בקצב אחיד, תיתכן עצירה פתאומית או פרץ נוסף בעת דילוג.
- **Zoom**- שליחה מהירה וקבועה של חבילות קטנות מאוד ללא הפסקות, עם שינוי קל בקצב בהתאם לדיבור או תנועה.
- **Spotify**- פרץ ראשוני קטן לטעינה מוקדמת, לאחר מכן שידור אחיד של חבילות קטנות עד בינוניות ללא שינויים משמעותיים.
- **Chrome**- פרץ גדול של חבילות בגודל משתנה בתחילת טעינת עמוד, ולאחר מכן האטה ניכרת עם קפיצות פתאומיות בזמן אינטראקציה.
- **Firefox**- דפוס דומה ל Chrome -אך עם פחות חבילות רקע, פרץ ראשוני חזק בטעינת דף ולאחר מכן תנועה בלתי סדירה, קפיצות חדות בעת אינטראקציה.