חלק א

.1

סוגי פרוטוקולים בשכבת התעבורה

ישנם שני פרוטוקולים מרכזיים המשמשים להעברת קבצים:

1. TCP אשר מבטיח העברה מסודרת ואמינה של הנתונים.
2. UDP אשר מיועד להעברה מהירה אך לא בהכרח אמינה

כל העברת קובץ תתחיל קודם משכבת האפליקציה, בה המשתמש מבצע פעולה כלשהי כמו העלאה לשרת או הורדת קובץ למשל באמצעות:

- תוכנות להעברת קבצים כמו FTP/SFTP
- הורדת קובץ מדפדפן HTTP/HTTPS
- גישה לקבצים ברשת המקומית SMB/NFS

שכבת האפליקציה מעבירה את הנתונים לשכבת התעבורה, אשר מחליטה כיצד לנהל את החיבור והעברת הנתונים בהתאם לפרוטוקול שנבחר.

תרחיש TCP

מכיוון TCP הוא פרוטוקול חיבורי (Connection-Oriented) יש צורך בהקמת חיבור לפני העברת נתונים לכן הוא תמיד יתחיל בלחיצת יד משולשת הכוללת 3 שלבים :

- SYN הלקוח שולח בקשת חיבור לשרת
- ACK-SYN השרת מאשר את הבקשה של הלקוח
- ACK הלקוח מאשר את הsequence של השרת

לאחר שלב זה החיבור מבוסס וניתן להתחיל לשלוח את הנתונים.

<u>בעיות שיכולות להיות בשלב זה -</u>

- **אובדן חבילות –** אם אחת מחבילות הSYN או הACK-SYN הולכת לאיבוד ,יש צורך בהמתנה לTimeout ושליחה מחדש.

- **RTT גבוה –** ככל שהמרחק בין הלקוח לשרת גדול יותר , זמן השהייה של ההתקשרות הראשונית יהיה ארוך יותר מכיוון שנדרש לעבור דרך יותר כבלי תקשורת אז זמן propagation יעלה

- **עומס ברשת –** אם השרת עמוס או שיש עומס בתורים של הנתבים התשובה ACK-SYN תגיע מאוחר יותר
למשל אם יותר מידי חיבורים חדשים נפתחים בבת אחת , השרת לא יקבל חיבורים חדשים עד שהתור יתפנה מה שיעכב את הקמת הקשר, יגרום ל timeout ולשליחה מחדש של בקשה להקמת קשר

לאחר שהחיבור מבוסס, TCP מחלק את הנתונים לסגמנטים כאשר כל סגמנט מורכב מ:

- **header** - מכיל מידע חשוב כמו Sequence Number, Acknowledgment Number, Flags, Checksum, וכו'.

- **Payload** - הנתונים עצמם ששכבת האפליקציה ביקשה לשלוח

במהלך לחיצת היד המשולשת כל צד בתקשורת מצהיר על MSS (Maximum Segment Size) , המגדיר את כמות הנתונים המקסימלית שסגמנט TCP יכול לשאת, ללא ה-Headers של TCP ו-IP.
אם שני הצדדים מצהירים על ערכי MSS שונים, החיבור ישתמש בערך הנמוך מביניהם כדי להבטיח תאימות ולמנוע בעיות במהלך ההעברה.

MSS מחושב על בסיס (Maximum Transmission Unit), שהוא גודל החבילה המקסימלי שניתן להעביר דרך הרשת. כאשר הערך שנבחר עבור MSS הוא גודל ה-MTU של הרשת בניכוי גודל ה-Headers של IP ו-TCP.

<u>השפעת גודל ה-MSS על ביצועי התקשורת:</u>

- **MSS קטן מדי** - אם כל סגמנט מכיל כמות קטנה של נתונים, נשלחות יותר חבילות, מה שגורם ליותר תקורה ומעמיס על הרשת. כל חבילה דורשת טיפול נוסף (כגון שליחה מחדש של ACK) ובסופו של דבר עשויה להוביל לניצול לא יעיל של רוחב הפס.

- **MSS גדול מדי** - אם ה-MSS גדול יותר מה-MTU של הרשת, ייתכן שהחבילות יחרגו מהמגבלות של הרשת ויתפצלו (Fragmentation). הדבר גורם לעיבוד נוסף ברשת ולסיכון לאיבוד חבילות, דבר שיגרום להאטה בהעברת הנתונים ויעלה את זמני השהייה ברשת.

שלב העברת הנתונים מתבצע תוך שמירה על אמינות וסדר הנתונים בעזרת מנגנונים המבטיחים שכל סגמנט יגיע ליעדו בצורה תקינה ובסדר הנכון.

<u>במהלך ההעברה TCP משתמש ב -</u>

- **מספור חבילות sequence number -** כל סגמנט מקבל רצף ייחודי.

- **אישורים Ack -** המקבל שולח חזרה הודעות ACK כדי שהשולח ידע שהנתונים התקבלו בהצלחה.

- **בקרת זרימה -** TCP – Flow control מתאים את קצב השידור שלו לפי יכולת העיבוד של המקבל על מנת למנוע עומס.

<u>מספור החבילות -</u>

אם חלק מהסגמנטים מגיעים בסדר לא נכון, המקבל מחכה לחבילות החסרות לפני שהוא שולח אישור ACK TCP אינו שולח חבילות חדשות מיד אלא מחכה לRetransmission Timeout מה שמוביל להשהיה. בנוסף אם סגמנטים נשלחים דרך מסלולים שונים ברשת, ייתכן מצב בו יגיעו בזמנים שונים מה שעלול לגרום להמתנה ושיחזור הסדר.

<u>אישורים ACK-</u>

TCP דורש אישור (ACK) עבור כל סגמנט כדי לוודא שהוא התקבל בהצלחה לפני המשך ההעברה.

- **RTT גבוה -** אם זמן הסיבוב של החבילה (RTT) ארוך, ייקח יותר זמן לקבל אישור עבור הנתונים שנשלחו, ולכן קצב ההעברה יקטן.

- **Delayed ACKs -** כאשר המקבל מחכה לפני שליחת אישור, קצב השידור של השולח יורד בהתאם.

- **אובדן חבילות ACK -** אם אישור אובד, השולח עשוי לחשוב שהנתונים לא התקבלו ולשלוח אותם מחדש, מה שמגביר את העומס על הרשת.

-Flow Control

בקרת זרימה נועדה לוודא שהשולח לא שולח יותר נתונים ממה שהמקבל יכול לעבד. TCP משתמש במנגנון Sliding Window, שבו השולח יכול לשלוח מספר סגמנטים ברצף מבלי לחכות לכל אישור בנפרד, כל עוד הוא נמצא בתוך גודל חלון הקבלה (Receive Window) שהמקבל מפרסם.

- **RTT גבוה -** כאשר זמן הסיבוב של חבילה (RTT) גדול, עדכון גודל חלון הקבלה מתבצע באיחור, והשולח עלול להמתין זמן רב יותר לפני שהוא מקבל אישור להמשיך לשלוח.
- **Receive Window קטן מדי -** אם היישום המקבל לא מספיק לקרוא נתונים מהר, הוא יקטין את ה-Receive Window, מה שיגרום לשולח לשלוח פחות נתונים בכל מחזור.
- **Zero Window (חלון סגור) -** אם המקבל מפסיק לחלוטין לפרסם חלון, השולח עוצר את השידור ושולח Zero Window Probes כדי לבדוק מתי אפשר להמשיך.
- **פערים בקצב השידור והקבלה -** אם קצב השידור מהיר מדי, המקבל עשוי לסגור את החלון, מה שיגרום להאטה משמעותית.

לאחר שלב העברת הנתונים, TCP משתמש במנגנון בקרת עומס כדי לוודא שהעברת הנתונים לא תגרום להצפה של הרשת. בעוד שבקרת זרימה (Flow Control) מתמקדת בקצב העיבוד של המקבל, בקרת עומס מתמקדת במצב הרשת עצמה.

TCP אינו מקבל מידע ישיר על מצב הרשת, ולכן הוא מעריך את רמת העומס על בסיס סימנים כמו איבוד חבילות או עיכובים בקבלת אישורים (ACKs).

המנגנון בנוי מארבעה שלבים עיקריים:

1. **Slow Start -** מתחיל לשלוח מעט נתונים ומגדיל בהדרגה את הקצב עד שמתגלה סימן לעומס.
2. **Congestion Avoidance -** מגביר את קצב השידור בצורה מתונה כדי למנוע קריסה ברשת.
3. **Fast Retransmit –** אם TCP מזהה אובדן של חבילות, הוא שולח אותן מחדש במהירות כדי למנוע האטה מיותרת.
4. **Fast Recovery -** מחזיר את קצב השידור למצב מאוזן לאחר זיהוי איבוד חבילות.

- TCP מתחיל את השידור במהירות נמוכה כדי לבדוק את מצב הרשת. במהלך שלב זה, קצב השידור עולה בהדרגה, מה שאומר שלוקח זמן עד שמגיעים למהירות שידור אופטימלית. אם החיבור הוא קצר (למשל בהעברת קובץ קטן).
  יתכן שעד שהתהליך מגיע למהירות המקסימלית, השידור כבר הסתיים, מה שמוביל לניצול נמוך של רוחב הפס.
- TCP מפרש אובדן חבילות כאות לכך שהרשת עמוסה. כאשר TCP מזהה שחבילה אבדה, הוא מקטין את קצב השידור באופן משמעותי כדי להפחית את הלחץ על הרשת. אך אם איבוד החבילות אינו נובע מעומס אלא מגורמים אחרים (כגון רשת לא יציבה), TCP מאט שלא לצורך, מה שגורם להשהיות מיותרות.

תרחיש UDP

מכיוון ש UDP לא מבצע בקרה על העברת הנתונים, הבעיות שגורמות להאטה כנראה ינבעו באופן ישיר מהרשת או מהאפליקציה עצמה.

- **עומס ברשת –**

UDP אינו מתאים את עצמו לעומס כמו TCP ולכן אם הרשת עמוסה חלק מהחבילות עלולות להיזרק. כאשר חבילות הולכות לאיבוד היישום יצטרך לשלוח אותן מחדש מה שייאט את ההעברה. ניתן לזהות בעיה זו כאשר חלק מהנתונים לא מגיעים כלל או שהם מגיעים בקצב משתנה.

- **איבוד חבילות –**

UDP תלוי ביציבות הרשת, אם חבילה הולכת לאיבוד היא נעלמת. יכול להיגרם מעומס בנתבים או למשל קריסות זמניות, רשת שאינה יציבה.

אם החבילות גדולות מידי ועוברות פרגמנטציה ייתכן שחלק מהחבילה ייאבד ואז כל החבילה לא תהיה שמישה מה שיוביל לבקשה חדשה.

על מנת לפתור את הבעיה נרצה לנסות להבין באיזה שלב היא מתרחשת:

ניטור חבילות הTCP בWireshark - המטרה היא לראות מידע אודות גודל חלון הקבלה. במידה והחלון קטן ייתכן שהמקבל אינו מסוגל לקבל נתונים במהירות מספקת ולכן החיבור מואט.
נחפש ACKים כפולים ושידורים חוזרים של חבילות שיצביעו לנו על זיהוי של איבוד חבילות או עומס מה שמקטין את קצב השידור.

3268	71.877769	192.168.1.135	192.168.1.56	TCP	164	[Continuation to #2025] [TCP Retransmission] 62670 → 8009 [PSH, ACK] Seq=1431 Ack=1431 Win=254 Len=110
5345	100.245393	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62763 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5346	100.245443	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62762 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5380	100.292771	2001:4860:4802:34::9d	2a00:a041:e9b8:5300:44cd:…	TCP	86	[TCP Dup ACK 5376#1] 443 → 62771 [ACK] Seq=1 Ack=1 Win=269312 Len=0 SLE=1441 SRE=1795
5407	100.450717	2001:4860:4802:34::9d	2a00:a041:e9b8:5300:44cd:…	TCP	86	[TCP Dup ACK 5404#1] 443 → 62771 [ACK] Seq=6859 Ack=2188 Win=267520 Len=0 SLE=3628 SRE=4040
5562	101.901321	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62777 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5563	101.901635	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62776 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5596	101.940284	2001:4860:4802:34::9d	2a00:a041:e9b8:5300:44cd:…	TCP	86	[TCP Dup ACK 5586#1] 443 → 62781 [ACK] Seq=1 Ack=1 Win=269312 Len=0 SLE=1441 SRE=1827
5632	102.165194	2001:4860:4802:34::9d	2a00:a041:e9b8:5300:44cd:…	TCP	86	[TCP Dup ACK 5625#1] 443 → 62781 [ACK] Seq=6859 Ack=2224 Win=267520 Len=0 SLE=3664 SRE=4018
5943	102.984502	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62783 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5944	102.984554	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62784 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
5945	102.984569	2a00:a041:e9b8:5300:44cd:…	2a00:a041:e9b8:5300:6aaa:…	TCP	86	[TCP Retransmission] 62782 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
14746	136.783990	2a00:a041:e9b8:5300:44cd:…	2a00:a040:0:6::d437:ba5b	TCP	74	[TCP Retransmission] 50430 → 80 [FIN, ACK] Seq=156 Ack=189 Win=65280 Len=0
14754	136.817013	2600:1901:1:5a5::	2a00:a041:e9b8:5300:44cd:…	TCP	86	[TCP Dup ACK 14751#1] 443 → 50431 [ACK] Seq=1 Ack=1 Win=269312 Len=0 SLE=1441 SRE=2092
14991	137.055359	2a00:a041:e9b8:5300:44cd:…	2600:1901:1:194::	TCP	74	[TCP Previous segment not captured] 60106 → 443 [ACK] Seq=216 Ack=19385 Win=255 Len=0
15331	137.367768	2a01:4ff:ef::fa57:1	2a00:a041:e9b8:5300:44cd:…	TLSv1.3	586	[TCP Previous segment not captured] , Continuation Data
15333	137.367833	2a00:a041:e9b8:5300:44cd:…	2a01:4ff:ef::fa57:1	TCP	86	[TCP Dup ACK 15310#1] 62775 → 443 [ACK] Seq=3949 Ack=8855580 Win=1047808 Len=0 SLE=9312988 SRE=9314012
15353	137.450039	2603:1020:805:3::401	2a00:a041:e9b8:5300:44cd:…	TCP	74	[TCP Dup ACK 2834#1] 443 → 49412 [ACK] Seq=152 Ack=2 Win=7720 Len=0
15362	137.510641	2a00:a041:e9b8:5300:44cd:…	2600:1901:1:194::	TCP	117	[TCP Retransmission] 60106 → 443 [PSH, ACK] Seq=173 Ack=83564 Win=255 Len=43
15386	137.651879	2a01:4ff:ef::fa57:1	2a00:a041:e9b8:5300:44cd:…	TCP	586	[TCP Retransmission] 443 → 62775 [ACK] Seq=8856508 Ack=3949 Win=65536 Len=512
15387	137.651879	2a01:4ff:ef::fa57:1	2a00:a041:e9b8:5300:44cd:…	TCP	586	[TCP Retransmission] 443 → 62775 [PSH, ACK] Seq=8857020 Ack=3949 Win=65536 Len=512
15388	137.651879	2a01:4ff:ef::fa57:1	2a00:a041:e9b8:5300:44cd:…	TCP	490	[TCP Retransmission] 443 → 62775 [ACK] Seq=8857532 Ack=3949 Win=65536 Len=416
15389	137.651879	2a01:4ff:ef::fa57:1	2a00:a041:e9b8:5300:44cd:…	TCP	586	[TCP Retransmission] 443 → 62775 [ACK] Seq=8857948 Ack=3949 Win=65536 Len=512

דוגמה לאיבוד חבילות:
בתמונה המצורפת, סימולצנו הורדת קובץ שנקטעה עקב Timeout. ניתן לראות מספר אינדיקטורים המעידים על בעיות בתעבורה:

- **TCP Retransmission** – שידור חוזר של חבילות בעקבות חוסר אישור מהמקבל.
- **Duplicate ACK** – אישורים כפולים המעידים על איבוד חבילה קודמת.
- **Previous segment not captured** – חבילה חסרה ברצף, המצביעה על אובדן מידע.

שימוש בs- netstat - הוא כלי סטטיסטי המאפשר ניתוח של פרוטוקולי התעבורה, ביניהם TCP ו-UDP, ומציג מידע מפורט על אופן התנהלות התעבורה ברשת. באמצעותו ניתן לאתר בעיות כגון אובדן חבילות, שידורים חוזרים , עומסים על החיבור , תקלות בתהליך האישור (ACKs) ונפילות חיבורים.

```
TCP Statistics for IPv4

  Active Opens                        = 73295
  Passive Opens                       = 15327
  Failed Connection Attempts          = 17623
  Reset Connections                   = 687
  Current Connections                 = 2
  Segments Received                   = 811112
  Segments Sent                       = 773766
  Segments Retransmitted              = 27408

TCP Statistics for IPv6

  Active Opens                        = 5093
  Passive Opens                       = 4
  Failed Connection Attempts          = 1584
  Reset Connections                   = 574
  Current Connections                 = 19
  Segments Received                   = 203174
  Segments Sent                       = 171429
  Segments Retransmitted              = 2591

UDP Statistics for IPv4

  Datagrams Received    = 413695
  No Ports              = 45225
  Receive Errors        = 0
  Datagrams Sent        = 74367

UDP Statistics for IPv6

  Datagrams Received    = 2763489
  No Ports              = 750
  Receive Errors        = 2
  Datagrams Sent        = 437218
```

- שידורים חוזרים (Segments Retransmitted) יכולים להצביע על איבוד חבילות.
- ניסיונות חיבור כושלים (Failed Connection Attempts) מעידים על בעיות תקשורת.
- מספר גבוה של Reset Connections עשוי להצביע על עומסים או חסימות חיבורים.
- Receive Errors ב-UDP עשוי לרמוז על בעיות באיכות הקישור.
- No Ports גבוה ב-UDP מצביע על חבילות שהגיעו ליעד לא תקין.

שימוש ב-P2P לחלוקת עומסים- ניתן להשתמש במנגנון P2P . בניגוד למודל Client-Server המסורתי, בו כל לקוח מקבל מידע ישירות מהשרת, במערכת P2P הלקוחות עצמם משתפים מידע זה עם זה. הדבר מאפשר הפחתת העומס על השרת המרכזי, שיפור קצבי ההעברה, והגדלת עמידות המערכת לאובדן חבילות.

יתרונות:

- הפחתת עומס על השרתים המרכזיים – מפחית צוואר בקבוק במערכת.
- שיפור יעילות העברת הנתונים – הלקוחות יכולים למשוך נתונים ממספר מקורות במקביל.
- עמידות טובה יותר לכשלים – גם אם שרת מרכזי כושל, הלקוחות יכולים להמשיך לשתף מידע.

מעבר לפרוטוקול QUIC כחלופה מתקדמת ל-TCP- QUIC הוא פרוטוקול תקשורת מודרני המבוסס על UDP, אשר פותח על ידי Google כחלופה מתקדמת ל-TCP. בעוד ש-TCP דורש תהליך שלם של Handshake ואישור חבילות לפני העברת נתונים, QUIC מבצע חיבור מהיר יותר ומקטין את הצורך בשידורים חוזרים (Retransmissions).

יתרונות של QUIC:

- זמן התחברות קצר יותר – הקמת חיבור TCP דורשת שלושה שלבים (SYN, SYN-ACK, ACK), בעוד QUIC מאפשר חיבור מידי.
- עמידות בפני אובדן חבילות – שימוש במנגנוני תיקון שגיאות פנימיים מפחית את הצורך בשידורים חוזרים ומגביר את יעילות השידור.
- אופטימיזציה לרשתות אלחוטיות – QUIC בנוי להתמודד עם השהיות ושינויים פתאומיים באיכות הרשת.

.2
מנגנון בקרת זרימה ב-TCP והשפעת עוצמת העיבוד על הביצועים

מנגנון בקרת זרימה (Flow Control) הוא מנגנון שנועד למנוע מהשולח להציף את המקבל בכמות נתונים גדולה מכפי שהוא מסוגל לעבד.

מניעת ההצפה -
במידה וההצפה אינה הייתה נמנעת על ידי מנגנון היינו יכולים להיתקל בתרחישים כמו:

- הצטברות חבילות עקב כך שהמקבל לא מסוגל לעבד במהירות מספקת, מה שיכול להוביל לזריקה של חבילות מצד המקבל ושליחה חוזרת מצד השולח מה שבסופו של דבר יאט אפילו יותר את ההעברה.
- המקבל עלול להפסיק להגיב עקב עומס, מה שיכול לגרום להאטה כללית של מערכת ההפעלה שלו, פגיעה גם בחיבורים אחרים שקיימים לו, להשעיה שלו ואפילו לניתוק.
- עומס על הזיכרון והמעבד של המקבל יגבירו את זמן השהיה ויגרמו לו לעבוד בצורה גרועה.

ב-TCP, בקרת הזרימה ממומשת באמצעות חלון קבלה (Receiver Window – rwnd), שבו המקבל מודיע לשולח בכל ACK כמה נתונים נוספים הוא מסוגל לקבל בכל רגע נתון.

השפעת עוצמת העיבוד:
כאשר עוצמת העיבוד של השולח גבוהה יותר, הוא מסוגל לעבד נתונים ולבצע חישובים במהירות גבוהה יותר, כולל שליחה וקבלה של חבילות נתונים. בפרוטוקול TCP, גם השולח וגם המקבל צריכים לעבד את חבילות הנתונים שהם מקבלים.

<u>כאשר השולח מהיר יותר מהמקבל:</u>
- השולח מסוגל ליצור ולשלוח חבילות במהירות גבוהה.
- המקבל, לעומת זאת, אינו מספיק לעבד את החבילות הנכנסות באותו קצב.
- כדי למנוע הצפה של המקבל, TCP משתמשת בבקרת זרימה.

איך בקרת הזרימה מאטה את השולח?

המקבל מעדכן את גודל החלון (rwnd) בהתאם לקצב שבו הוא מסוגל לעבד את הנתונים.
אם המקבל איטי, חלון הקבלה (rwnd) יהיה קטן יותר מגודל חלון השליחה של השולח, מה שיגרום לשולח להאט את קצב השליחה.
אם המקבל נמצא תחת עומס כבד או שאינו מצליח לעבד חבילות נוספות, הוא עשוי להקטין עוד יותר את ה-rwnd.
במקרים קיצוניים, אם המקבל לא מעדכן את ה-rwnd (למשל, אם הוא קורס או עמוס מאוד), השולח עשוי להפסיק לשלוח נתונים זמנית עד שיקבל אישור להמשך.

<u>השפעה על ניצול רוחב הפס:</u>
כאשר יש פער משמעותי בין עוצמת העיבוד של השולח למקבל, במקום שניצול רוחב הפס יהיה גבוה בשל הביצועים המהירים של השולח, העברת הנתונים הופכת לאיטית כי המקבל מהווה "צוואר בקבוק".
כלומר, מהירות ההעברה בפועל לא נקבעת לפי יכולות השולח, אלא לפי היכולת של המקבל להתמודד עם הנתונים.

.3

מהו ניתוב ?

ניתוב הוא התהליך שבו נתבים מחליטים לאיזה מסלול לשלוח חבילה מהרשת המקורית אל היעד.
הנתבים משתמשים בטבלאות ניתוב כדי לבחור את הנתיב המתאים ביותר בהתבסס על כתובת ה-IP של היעד.

סוגי ניתוב:

- **ניתוב סטטי**– נתיבים מוגדרים ידנית, ללא שינוי אוטומטי במקרה של כשל.
- **ניתוב דינמי**– הנתב מחשב את הנתיב האופטימלי באופן אוטומטי על פי פרוטוקולי ניתוב.
- **ניתוב ברירת מחדל**– חבילות נשלחות לנתיב ברירת מחדל אם אין התאמה לכללים אחרים.

כיצד בחירת המסלול משפיעה על ביצועי הרשת?

כאשר קיימים מספר מסלולים בין המקור ליעד, ישנם מספר גורמים מרכזיים שמשפיעים על הביצועים:

- **זמן השהיה** – משך הזמן שלוקח לחבילה להגיע ליעדה.
  זמן ההשהיה גדל כאשר מספר הנתבים בדרך גדול יותר, מכיוון שכל נתב מוסיף עיבוד נוסף של החבילה, המתנה בתור לפני שליחה, ובמקרים של עומסים – ייתכנו עיכובים משמעותיים נוספים.

- **רוחב פס** – הקיבולת המרבית של הנתיב להעברת נתונים.
  רוחב פס נמוך משפיע לרעה על הביצועים מכיוון שמספר מוגבל של חבילות יכול לעבור בו זמנית, מה שמוביל לעיכובים ולצווארי בקבוק.

- **אובדן חבילות** – מצב שבו חבילות מידע אינן מגיעות ליעדן.
  אובדן חבילות מתרחש כאשר הנתיב שנבחר עמוס, מה שגורם למחיקת חבילות עקב חוסר מקום בתור הנתב או כאשר חבילות נופלות עקב שגיאות בתקשורת.

- **שרידות** – היכולת של הרשת להתמודד עם כשל באחד הנתבים.
  מסלול ניתוב שאינו גמיש מספיק עלול לגרום לניתוקים במקרה של כשל, ולכן חשוב להגדיר מסלולים חלופיים.

- **איזון עומסים** – חלוקת התעבורה בין מספר נתיבים כדי למנוע עומסים.
  אם כל התעבורה מתנקזת לנתיב אחד בלבד, הוא עלול להפוך לעמוס, ולכן כדאי להשתמש באיזון עומסים כדי להפחית עומס מנתיבים מסוימים ולשפר את הביצועים הכלליים של הרשת.

מהם השיקולים בבחירת מסלול ניתוב אופטימלי?

מספר הנתבים בדרך (Hops)

ככל שמספר הנתבים בדרך גדול יותר, כך זמן ההשהיה גדל. כל נתב בדרך דורש עיבוד נוסף של החבילה, גורם להמתנה בתור לפני שליחתה, ועלול להגדיל את הסיכון לאובדן חבילות.

עלות הנתיב
עלות הנתיב היא מדד שמשמש פרוטוקולי ניתוב כדי להשוות בין מסלולים ולבחור את היעיל ביותר. פרוטוקולים כמו Open Shortest Path First מחשבים את עלות הנתיב לפי רוחב הפס, עומסים קיימים, ומספר הנתבים בדרך. הנתיב עם העלות הנמוכה ביותר נבחר.

עומס על הנתיבים
עומס על נתיב עלול לגרום לאובדן חבילות ולעיכובים משמעותיים. עומס מתרחש כאשר מספר החבילות שנכנסות לנתב גבוה מכושר השידור שלו.

רוחב פס
רוחב פס גבוה יותר מאפשר העברת מידע מהירה יותר ומקטין צווארי בקבוק. אם נתיב מסוים מוגבל ברוחב הפס שלו, הוא עלול לגרום להאטת קצב ההעברה של הנתונים.

אמינות המסלול
נתיבים מסוימים עלולים להיות פחות אמינים ולגרום לניתוקים תכופים או לאובדן חבילות. בבחירת מסלול יש להעדיף נתיבים יציבים יותר כדי למנוע הפרעות בתקשורת.

איזון עומסים
איזון עומסים מאפשר להפיץ את התעבורה בין מספר מסלולים כדי להקטין עומסים על נתיב מסוים. פרוטוקולים כמו EIGRP מאפשרים איזון עומסים דינמי, כך שהתעבורה תחולק לפי קיבולת הנתיבים.

איך פרוטוקולי ניתוב מחליטים על המסלול ?

פרוטוקולי ניתוב פנים-ארגוניים (IGP - Interior Gateway Protocols)
משמשים לניהול תעבורה בתוך רשתות פרטיות או ארגוניות.

- Routing Information Protocol – מחשב מסלול לפי מספר הנתבים בלבד, ללא התחשבות בעומסים או רוחב הפס.
- Open Shortest Path First – מחשב את המסלול הקצר ביותר בהתחשב בעלות הנתיב, תוך התחשבות ברוחב הפס ובגורמים נוספים.
- Enhanced Interior Gateway Routing Protocol – משתמש במספר פרמטרים כמו רוחב פס, זמן השהיה, ואמינות כדי לקבוע את הנתיב האופטימלי.

פרוטוקולי ניתוב חוץ-ארגוניים (EGP - Exterior Gateway Protocols)
משמשים לניהול תעבורה בין רשתות שונות, כגון אינטרנט.

- Border Gateway Protocol – מנהל את הנתיבים בין ספקי אינטרנט (ISP) ומתחשב במדיניות עסקית ובקריטריונים טכניים לבחירת הנתיב.

איך לבחור מסלול ניתוב נכון ?
ישנם מספר שיקולים מרכזיים שחשוב לקחת בחשבון בעת בחירת מסלול ניתוב כדי להבטיח ביצועים טובים של הרשת:

- בחירת נתיבים עם **רוחב פס גבוה** כדי למנוע צווארי בקבוק ולשפר את קצב ההעברה.
- צמצום **מספר הנתבים בדרך** כדי להפחית השהיות ולמנוע עיבוד יתר של החבילות.
- הימנעות מנתיבים עם **עומס גבוה**, שכן נתיבים עמוסים עלולים לגרום לאובדן חבילות ולעיכובים.

- שימוש בפרוטוקולים חכמים כמו OSPF , המאפשרים חישוב דינמי של המסלולים האופטימליים בהתבסס על נתוני הרשת.
- הפעלת איזון עומסים דינמי, המאפשר פיזור של תעבורה על פני מספר מסלולים כדי לשפר את הביצועים ולהימנע מעומסים נקודתיים.

בחירה נכונה של מסלול הניתוב תשפר את הביצועים של הרשת, תקטין את זמני ההשהיה, ותבטיח חיבור יציב ויעיל לאורך זמן.

4.
פרוטוקול MPTCP –

Multipath TCP הוא הרחבה של פרוטוקול TCP המאפשרת שימוש במספר מסלולים במקביל להעברת נתונים בין מקור ליעד. בניגוד ל-TCP רגיל, אשר פועל על בסיס מסלול יחיד, MPTCP מנצל נתיבים מרובים בו-זמנית, ובכך משפר את ביצועי הרשת בהיבטים של הפחתת זמן השהיה, יתירות, רוחב פס ואיזון עומסים.

- **הפחתת זמן השהיה ושיפור ניצול הרשת**
TCP רגיל מוגבל למסלול יחיד, ולכן כאשר נתיב מסוים עמוס, כל התעבורה חייבת לעבור דרכו, דבר המוביל לעלייה בזמני ההשהייה ולירידה ביעילות הרשת. MPTCP מאפשר פיצול התעבורה בין מספר מסלולים במקביל, מפחית עומסים ומשפר את מהירות ההעברה וניצול המשאבים.

- **שרידות גבוהה ועמידות בפני נפילות רשת**
ב-TCP רגיל, כשל במסלול מוביל לשבירת החיבור ולצורך ביצירת חיבור חדש. תהליך זה גורם לשיבושים בתקשורת ולירידה בזמינות הרשת. לעומת זאת, MPTCP מספק יתירות, כך שגם במקרה של כשל באחד המסלולים, הנתונים ממשיכים לעבור דרך המסלולים הנותרים ללא הפרעה.

- **הגדלת רוחב הפס ושיפור קצב ההעברה**
TCP רגיל מוגבל לרוחב הפס של מסלול יחיד, ולכן מהירות ההעברה תלויה בקיבולת הנתיב הבודד. לעומת זאת, MPTCP מאפשר שילוב של מספר חיבורים שונים במקביל, מה שמאפשר ניצול טוב יותר של רוחב הפס הכולל ומגביר את קצב ההעברה.

- **איזון עומסים (Load Balancing)**
ב-TCP רגיל, כל התעבורה עוברת דרך מסלול יחיד, מה שעלול לגרום לצווארי בקבוק ברשת כאשר הנתיב עמוס. MPTCP מזהה עומסים ומפצל את הנתונים בין מספר מסלולים, ובכך מאפשר חלוקה חכמה של העומס בין הנתיבים, משפר את ביצועי הרשת ומונע האטות כתוצאה מעומס יתר.

השוואה בין MPTCP ל-TCP רגיל

מאפיין	MPTCP	TCP רגיל
מסלול תקשורת	מספר מסלולים במקביל	נתיב יחיד בלבד
התאוששות מכשל	הנתונים ממשיכים לעבור דרך מסלולים חלופיים	נדרש חיבור מחדש במקרה של נפילה

רוחב פס	משלב מספר קישורים להגדלת קצב ההעברה	מוגבל למהירות של קו אחד
איזון עומסים	מעביר תנועה ממסלול עמוס למסלול פנוי	אין
גמישות והתאמה דינאמית	מתאים את עצמו לשינויים באופן אוטומטי	לא מתאים לשינויים בתנאי הרשת

MPTCP מציע שיפורים משמעותיים בהשוואה ל-TCP רגיל, בזכות היכולת לנצל מסלולים מרובים. יתרונותיו המרכזיים באים לידי ביטוי בשיפור הביצועים, הגדלת היציבות והרציפות של החיבור, העלאת קצב ההעברה וייעול השימוש במשאבי הרשת. יכולות אלו הופכות את MPTCP לפרוטוקול אידיאלי עבור סביבות עבודה דינאמיות ורשתות בעלות עומסים משתנים.

.5

תהליך מעבר חבילות בין נתבים

כאשר המחשב שולח נתונים, המידע מחולק לחבילות קטנות יותר בהתאם לפרוטוקול ההעברה (למשל, TCP או UDP). החבילות עוברות דרך נתבים שונים לאורך המסלול ליעדן. כל נתב משתמש בטבלת ניתוב כדי להחליט לאן להעביר את החבילה הבאה. לאורך הדרך, יתכנו בעיות שונות שעלולות לגרום לאובדן החבילות.

גורמים אפשריים לאובדן חבילות ודרכי פתרון

1. עומס על הנתב

נתבים משתמשים בתורים פנימיים (Buffers) לאחסון חבילות לפני שהן מועברות הלאה. אם הנתב מקבל חבילות בקצב גבוה יותר משהוא מסוגל לשדר, התורים מתמלאים, והנתב מתחיל להפיל חבילות חדשות כדי לשמור על יציבות הפעולה.

גורמים אפשריים לעומס

- **עומס יתר שנגרם מתעבורה גבוהה** – הנתב מחלק את רוחב הפס בין כל המכשירים ברשת. כאשר מתבצעות הורדות כבדות (למשל סטרימינג, העברת קבצים גדולים, עדכונים), הנתב צריך לטפל בכמות גדולה של חבילות בבת אחת. אם רוחב הפס מוגבל או שיש ריבוי מכשירים פעילים, העומס עלול לגרום לאובדן חבילות.
- **ריבוי חיבורים במקביל** – למשל, תוכנות P2P יוצרות חיבורים מרובים בו-זמנית, מה שמגדיל את העומס על הנתב.
- **ניהול NAT** – נתבים ביתיים משתמשים ב-NAT (תרגום כתובות רשת) לניהול חיבורים חיצוניים. כאשר מספר החיבורים גדול, נדרש עיבוד נוסף, שעשוי להוביל לאובדן חבילות.
- **איכות הנתב** – נתבים ישנים או בעלי חומרה חלשה (מעבד איטי, זיכרון נמוך) מתקשים לטפל בתעבורה גבוהה.
- **חוסר בניהול עומסים (QoS)** – ללא הקצאת עדיפויות לתעבורה קריטית, הנתב עלול להפיל חבילות חשובות.
- **פערי קצב העברה בין נתבים** – חיבור נתב מהיר לנתב איטי עלול לגרום לאובדן חבילות בשל אי התאמה במהירויות.

פתרונות אפשריים

- **שימוש ב-QoS** – תעדוף סוגים מסוימים של תעבורה (כגון שיחות וידאו) על פני תעבורה אחרת (כגון הורדות).
- **הגבלת רוחב פס להורדות כבדות** – ניתן להגדיר זאת דרך הנתב או תוכנות ייעודיות לניהול רשת.
- **שדרוג נתב** – נתבים מתקדמים עם חומרה טובה יותר יכולים להתמודד עם תעבורה כבדה בצורה יעילה.
- **איזון עומסים**– שימוש במספר חיבורים לאינטרנט או התקני איזון עומסים חכמים יכול למנוע עומס יתר על נתב אחד.

2. בעיות בניתוב

נתבים מחליטים על כיוון שליחת החבילות בהתבסס על טבלת הניתוב שלהם. בעיות בטבלה זו עלולות לגרום ל:

- **שליחת חבילות לכתובת שגויה**
- **יצירת לולאות ניתוב (Routing Loop)** – מצב שבו חבילות מסתובבות בין נתבים מבלי להגיע ליעדן
- **השלכת חבילות (Packet Drop)** – כאשר הנתב אינו מזהה נתיב תקף להעברה

גורמים לבעיות בניתוב

- **הגדרות שגויות בטבלאות ניתוב ידניות** – טעויות אנוש עשויות לגרום לחבילות שלא להגיע ליעדן.
- **כשל בפרוטוקולי ניתוב דינמיים (OSPF, BGP, RIP)** – מידע שגוי יכול לגרום לנתבים לבחור מסלולים לא תקינים.
- **רשת חדשה שלא הוגדרה נכון** – חבילות עלולות להיזרק אם הנתבים לא מעודכנים לגבי הרשת החדשה.
- **תקלה בנתב קריטי** – נתב תקול עלול להמשיך לקבל חבילות ולגרום ללולאת ניתוב או איבוד נתונים.
-

פתרונות אפשריים

- **בדיקות ועדכון טבלאות הניתוב באופן קבוע** – לוודא שההגדרות מעודכנות, במיוחד במערכות שבהן הניתוב נעשה ידנית
- **הגדרת ברירת מחדל נכונה** – להגדיר נתיב ברירת מחדל תקין במקרה שאין התאמה לטבלאות הניתוב הקיימות.
- **שימוש בפרוטוקולי ניתוב עם זיהוי כשלים** – למשל, פרוטוקולים כמו OSPF תומכים בזיהוי מהיר של נתבים כושלים ומעדכנים את הנתיבים בהתאם.
- **שימוש בניתוב סטטי פשוט** – במקרים מסוימים עדיף להגדיר ניתוב סטטי אם הרשת קטנה ויציבה, כדי למנוע שגיאות שעלולות להיגרם מפרוטוקולים דינמיים.
  החיסרון - אם נתב מסוים נופל, המידע לא יתעדכן אוטומטית כמו בפרוטוקולים דינמיים.

- **הפעלה מחדש של הנתב במקרה של בעיות** – לעיתים, תקלות נגרמות בגלל זיכרון עמוס או תקלות תוכנה, והפעלה מחדש של הנתב עשויה לפתור אותן.
נתבים משתמשים בזיכרון RAM לאחסון טבלאות ניתוב. אם הזיכרון מתמלא או יש תקלה, חבילות עלולות ללכת לאיבוד.

3. TTL נמוך (Time-To-Live)

TTL הוא ערך מספרי המתווסף לכל חבילה ונועד למנוע מחבילות לשוטט לנצח ברשת. בכל מעבר חבילה דרך נתב, ה-TTL קטן ב-1. אם TTL מגיע ל-0, הנתב זורק את החבילה ולא משדר אותה הלאה.

גורמים להורדת TTL מוקדמת

- **הגדרת TTL נמוכה מדי בתחילת המסלול-** חלק מהמערכות שולחות חבילות עם TTL נמוך מדי מלכתחילה מה שגורם לחבילות 'למות' לפני שהגיעו אל היעד. דוגמה למערכות כאלו הן מערכות שמשתמשות במגבלות אבטחה אשר מגבילות את הTTL כדי למנוע יציאה של חבילות לרשת החיצונית.
מקרה נוסף הוא כאשר חבילה נשלחת מהרשת המקומית (LAN) עם TTL נמוך , היא עלולה להיכשל במידה והיעד נמצא ברשת הרחבה (WAN)
- מסלול ארוך מהצפוי עם מספר גדול של נתבים -לולאות ניתוב שגורמות לחבילה להסתובב עד שה-TTL שלה פג - יכול להיגרם עקב טעות בטבלאות הניתוב , עדכון שגוי בפרוטוקולי הניתוב הדינמיים.

פתרונות אפשריים

- **נבדוק את הRTT** – נשלח ping לשרת היעד ונבדוק מה זמן התגובה והאם הוא גבוה. בנוסף ניתן לשלוח פינג רציף בשביל לראות אם יש איבוד של חבילות .
- **בדיקת המסלול על ידי שימוש ב traceroute -** כדי לבדוק כמה נתבים החבילות עוברות . אם יש נתב מסוים עם זמן תגובה גבוה או איבוד חבילות ייתכן שזה הגורם להאטה. אם המסלול עובר דרך הרבה תחנות ביניים זה יכול לגרום להשהיות נוספות.

4. בעיות MTU (Maximum Transmission Unit)

MTU קובע את הגודל המקסימלי של חבילה שניתן להעביר ברשת. אם חבילה גדולה מה-MTU של אחד הנתבים בדרך, היא צריכה לעבור **פרגמנטציה (Fragmentation)**. אם פרגמנטציה לא מתבצעת מסיבה כלשהי (למשל, אם דגל "Don't Fragment" מופעל), החבילה תיזרק.

למה זה קורה?

- הבדל בין ערכי MTU לאורך המסלול- רשתות שונות משתמשות בערכי MTU שונים.
- חסימה של פרגמנטציה במדיניות האבטחה של הרשת- אם חבילה גדולה מדי וצריכה לעבור פיצול אך הפרגמנטציה חסומה , הנתב יזרוק את החבילה במקום לשלוח אותה כמקטעים קטנים
- שרתים ששולחים חבילות גדולות מדי ללא התאמה לרשת.

השפעת אובדן חבילות על TCP ו-UDP

אובדן חבילות משפיע באופן שונה על TCP ו-UDP בשל אופן ניהול התקשורת שלהם. בעוד TCP כולל מנגנוני תיקון שגיאות והתאמה דינמית של קצב השידור, UDP מתמקד בשידור מהיר ללא בקרת אמינות, מה שהופך אותו לרגיש יותר לאובדן חבילות.

השפעת אובדן חבילות על TCP

TCP הוא פרוטוקול אמין המתבסס על בקרת שגיאות ושידור חוזר. כאשר חבילה הולכת לאיבוד, TCP מזהה זאת באמצעות היעדר אישור קבלה (ACK) או חוסר תגובה במסגרת הזמן המוגדרת (Timeout). כתוצאה מכך, החבילה נשלחת מחדש. עם זאת, כאשר שיעור אובדן החבילות גבוה, TCP מפעיל בקרת עומס (Congestion Control) ומקטין את החלון, מה שמוביל להאטת קצב השידור כדי למנוע עומס נוסף.
בנוסף, TCP מודד את זמן ההגעה-חזרה (RTT) של החבילות כדי להעריך את מצב הרשת. אם RTT עולה באופן משמעותי, TCP מפרש זאת כסימן לעומס ומפחית את קצב השידור בהתאם. בקרת הזרימה (Flow Control) מסייעת להתאים את קצב השידור ליכולת הקבלה של הצד המרוחק, אך אם קיבולת הנתב נמוכה או שהעומס ברשת גבוה, החבילות עשויות להיזרק, וכתוצאה מכך לגרום להאטה נוספת.

השפעת אובדן חבילות על UDP

UDP הוא פרוטוקול חסר חיבור (Connectionless), כלומר הוא אינו מבצע תיקון שגיאות מובנה או שידור חוזר של חבילות אבודות. כאשר חבילה הולכת לאיבוד, היא אינה משוחזרת אלא אם היישום עצמו מיישם מנגנון תיקון שגיאות חיצוני. לכן, שירותים מבוססי UDP כמו למשל סטרימינג ומשחקים מקוונים רגישים במיוחד לאובדן חבילות, שכן הם מסתמכים על שידור רציף ומהיר ואינם יכולים להמתין לשידור חוזר של חבילות. במקרים של עומס ברשת, נתבים עשויים להפיל חבילות UDP, מה שעלול להוביל לאיכות קול ירודה, פיקסול בסטרימינג או השהיות בלתי צפויות במשחקים מקוונים.

למה זה קורה?

עומס ברשת
כאשר נתבים מוצפים בחבילות, הם עשויים להפיל חבילות כדי למנוע עומס יתר. TCP יזהה את האובדן ויאט את השידור, אך UDP ימשיך לשלוח נתונים ללא תיקון.

RTT גבוה
אם ה-RTT גבוה מדי, TCP יקטין את קצב השידור כדי למנוע עומס נוסף. UDP, שאינו מבצע ניתוח RTT, ימשיך לשדר בקצב קבוע גם אם חלק מהחבילות נעלמות.

חוסר התאמה בין קצב השידור לקיבולת הרשת
TCP מוודא שהצד המקבל יכול להתמודד עם קצב השידור, אך אם הרשת עמוסה, קצב ההעברה יקטן.
UDP אינו מבצע התאמה דינמית, ולכן במקרה של עומס, החבילות פשוט יאבדו.

גודל חבילות גדול מה-MTU
אם חבילה גדולה מה-MTU של אחד הנתבים במסלול ואין אפשרות לפרגמנטציה, החבילה תיזרק.

פתרונות אפשריים -

עבור TCP

- שימוש בפרוטוקולי תעבורה מבוזרים כמו MPTCP (Multipath TCP) כדי לפצל תעבורה על פני מסלולים מרובים.
- הפעלת מנגנוני תיקון שגיאות מתקדמים כמו SACK (Selective Acknowledgment) לשיפור ביצועים בתנאים של אובדן חבילות - במקום לשדר מחדש הכל, המקבל מדווח אילו חבילות הגיעו בשלמותן ואילו חסרות. השולח מבצע שידור חוזר רק לחבילות החסרות, מה שחוסך זמן, משאבים

ורוחב פס.
ברוב הרשתות המודרניות SACK מופעל כברירת מחדל, אך ברשתות ישנות או מוגבלות ייתכן שעדיין משתמשים במנגנון הישן של שידור מחדש לכל החבילות.

עבור UDP

- הפעלת QoS כדי לתת עדיפות לתעבורה קריטית כמו שיחות VoIP או סטרימינג.
- הקטנת העומס ברשת על ידי ויסות עומסי נתונים או הגבלת קצב ההורדות ברקע.

שימוש ב WireShark למציאת הבעיה

wireshark מאפשר לזהות ולנתח אובדן חבילות בשכבת התעבורה על ידי מעקב אחר תעבורת TCP ו-UDP ובחינת דפוסי השידור, כגון שידורים חוזרים, אישורים כפולים וחבילות חסרות. לאחר זיהוי הבעיה, ניתן לנקוט בצעדים מתאימים לשיפור ביצועי הרשת.

- אם Wireshark מציג ריבוי של שידורים חוזרים, זה מעיד על אובדן חבילות, מה שעלול לגרום להאטה בהעברת הנתונים.
- אם נמצאות חבילות שמראות שהחלון המוגדר קטן מדי, זה מצביע על כך שהרשת מפחיתה את קצב השידור כדי למנוע עומס נוסף.
- RTT גבוה במיוחד, ייתכן כי אובדן חבילות נגרם עקב עומס יתר או תקלות בנתיבים ברשת.
- זיהוי חבילות גדולות שלא עברו כראוי, ייתכן שה-MTU לאורך הנתיב נמוך מדי או שהפרגמנטציה חסומה.

חלק ב

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

התרומה העיקרית של המאמר

המאמר מציג את FlowPic, גישה חדשנית לסיווג תעבורת אינטרנט מוצפנת. בשונה משיטות מסורתיות המבוססות על חילוץ תכונות ידני ולמידת מכונה רדודה, FlowPic ממיר נתוני זרימה לרשת לתמונות ומשתמש ברשתות נוירונים קונבולוציוניות (CNNs) לצורך הסיווג. השיטה משיגה תוצאות פורצות דרך, במיוחד בזיהוי תעבורה מוצפנת (VPN, Tor), ומהווה חלופה מדויקת ויעילה בהשוואה לגישות קיימות.

היתרונות המרכזיים של השיטה:

- **דיוק גבוה** - המודל מזהה תעבורה ביעילות, גם עבור נתונים מוצפנים כמו VPN ו-Tor, עם דיוק שמגיע ליותר מ-99.7% עבור זיהוי אפליקציות.

- **עמידות להצפנה** - הגישה אינה תלויה בתוכן המטען של החבילות (payload), ולכן היא מסוגלת לזהות תעבורה מוצפנת מבלי לפגוע בפרטיות. הדרישה לאחסון היא מינימלית מאוד ועבור על חבילה יש צורך להעביר רק שתי מילים של נתונים מהמנוע המעביר אל מנגנון הניתוח מה שהופך את הסיווג בזמן אמת לאפשרי

- **הכללה טובה** - ניתן לזהות אפליקציות חדשות שלא נכללו בשלבי האימון, כלומר השיטה מבוססת על מאפייני הקטגוריה עצמה ולא רק על אפליקציות מסוימות.
גישה גנרית לסיווג תעבורת אינטרנט המנצלת את על המידע הזמין על זמן וגודל חבילות ברשת במקום להסתמך על תכונת שנבחרו באופן ידני

- **דרישות נמוכות לאחסון ולחישוב** - אין צורך לשמור כמויות גדולות של נתונים, כי השיטה מסתמכת רק על זמני הגעת החבילות והגדלים שלהן.

מאפייני תעבורה במאמר

בשונה משיטות מסורתיות, FlowPic משתמש רק בתכונות הקשורות לגודל וזמן חבילה, מה שהופך אותו ליעיל ושומר פרטיות.

ייצוג FlowPic:

- ציר X – זמן הגעת החבילה (מנורמל)
- ציר Y – גודל החבילה (מוגבל ל-1500 בתים, MTU של Ethernet)
- עוצמת הפיקסל – תדירות החבילות של גודל מסוים בזמן מסוים

שימוש ב-CNNs לסיווג תעבורה:

- בניגוד לשיטות סטטיסטיות מסורתיות או מודלים מבוססי תכונות ידניות, FlowPic מנצל למידה עמוקה מעולם זיהוי התמונות.

עמידות בפני הצפנה:

- FlowPic לוכד מאפייני תעבורה אינהרנטיים, ולכן אינו רגיש להצפנה ב-VPN ו-Tor.

יעילות באחסון ובחישוב:

- מצריך שני ערכים בלבד לכל חבילה (גודל וזמן), מה שהופך אותו למתאים לסיווג בזמן אמת.

תכונות ייחודיות של השיטה:

- **שימוש בעיבוד תמונה לצורך סיווג תעבורה** - גישה חדשנית זו מאפשרת למודל לזהות דפוסים שהתגלו בעיבוד תמונות ולא נצפו בעבר בתעבורה מוצפנת.
- **אינה דורשת בחירת תכונות ידנית** - בניגוד לשיטות מסורתיות שמתבססות על בחירת תכונות סטטיסטיות ידניות, המודל מזהה את הדפוסים בעצמו באמצעות רשת נוירונים.
- **ניתן לאימון עם תעבורה שאינה מוצפנת ועדיין לזהות תעבורה מוצפנת** - FlowPic מאפשר לאמן את הרשת על זרימות רגילות ולזהות במדויק זרימות VPN ו-Tor.

הממצאים העיקריים והתובנות

המאמר מציג סדרת ניסויים המוכיחים את יעילות השיטה. עיקרי הממצאים:

משימת סיווג	דיוק FlowPic	תוצאה קודמת
קטגוריזציה – Non-VPN	85%	84%
קטגוריזציה – VPN	98.4%	98.6%
קטגוריזציה – Tor	67.8%	84.3%
Non-VPN Class vs. All	97%	-
VPN Class vs. All	99.7%	-
Tor Class vs. All	85.7%	-
סיווג שיטות הצפנה	88.4%	99%
זיהוי יישומים	99.7%	93.9%

FlowPic מציג ביצועים מעולים, במיוחד בזיהוי יישומים (99.7% דיוק).

Tor נותר האתגר הגדול ביותר לסיווג תעבורה, עם דיוק נמוך יחסית (67.8%).

ביצועי Class vs. All לפי הצפנה

FlowPic נבדק על Non-VPN, VPN ו-Tor, ומציג תוצאות של:

- זיהוי VoIP – 99.6% דיוק ב-Non-VPN, 99.9% ב-VPN, אך רק 48.2% ב-Tor.
- זיהוי וידאו – 99.9% ב-Non-VPN ו-VPN, אך 83.8% ב-Tor.
- זיהוי צ'אט וגלישה – תוצאות נמוכות יותר אך עדיין מדויקות.

VPN אינו פוגע משמעותית בדיוק הסיווג (~98.4%), אך Tor מקשה מאוד על הסיווג.

הכללה ליישומים לא מוכרים

- גם כאשר Facebook Video לא נכלל באימון, FlowPic סיווג אותו נכון ב-99.9%.
- גם ללא Vimeo ו-YouTube, דיוק הזיהוי נשמר ב-83.1%.

FlowPic מזהה מאפיינים כלליים של קטגוריות תעבורה, ולא מסתמך רק על דפוסים של יישומים ספציפיים.

תובנות מרכזיות

- למידה עמוקה מבוססת תמונות היא גישה פורצת דרך לסיווג תעבורה.
- VPN פחות משפיע על הדיוק לעומת Tor, כנראה בשל ההבדלים בטכניקות ההצפנה.
- FlowPic מתאים ליישומים חדשים, ומציג פוטנציאל לפריסה בזמן אמת.
- CNNs יכולים ללמוד התנהגות תעבורה מהותית ללא צורך במודלים מותאמים אישית.

מסקנות

- FlowPic ממיר סיווג תעבורה מוצפנת למשימת זיהוי תמונות, ומשיג דיוקים גבוהים במיוחד.
- דיוק גבוה בתעבורת(~98.4%) VPN מוכיח עמידות גבוהה בפני הצפנה.
- Tor עדיין מאתגר לסיווג (~67.8%), בשל רמות טשטוש נתונים גבוהות יותר.
- זיהוי היישומים כמעט מושלם (~99.7%), מה שמעיד על למידת מאפייני תעבורה כלליים.
- השיטה יעילה ושומרת על פרטיות, שכן אין צורך לנתח את תוכן המנות (Payload).

FlowPic משנה את כללי המשחק ומוכיח שסיווג תעבורה יכול להיות פשוט כמו זיהוי תמונות

אופק בר שלום - 324161421 **ג'סי בריגס** - 314708041 **עמית דנינו** - 323084434 **רועי נאור** - 323917104

Early Traffic Classification With Encrypted ClientHello: A Multi-Country Study

התרומה העיקרית של המאמר

המאמר מציע שיטה חדשה לסיווג מוקדם של תעבורה מוצפנת (eTC), בעיקר עבור TLS 1.3 עם Encrypted ClientHello.

בעוד שהצפנת ECH מסתירה מידע קריטי כגון SNI, המאמר מציג אלגוריתם חדש בשם Hybrid Random Forest Traffic Classifier אשר מנצל נתונים בלתי מוצפנים הנותרים ב-TLS handshake, לצד מאפיינים סטטיסטיים של זרימת התעבורה, כדי לשפר את יכולת הסיווג.

<u>תרומת המאמר באה לידי ביטוי בכמה היבטים מרכזיים:</u>

- **שימוש בתכונות בלתי מוצפנות ב-TLS handshake** – האלגוריתם משתמש במאפיינים שנותרו זמינים, כמו גרסת הפרוטוקול, קבוצות מפתחות הצפנה, ואורך ההרחבות.
- **איסוף מאגר נתונים רחב** – נבנה מאגר נתונים הכולל מעל 600,000 זרימות TLS ב-19 קטגוריות שונות, אשר נאספו ממדינות שונות בצפון אמריקה, אירופה ואסיה.
- **שיפור ביצועים משמעותי** – האלגוריתם hRFTC משיג דיוק של עד 94.6% בסיווג תעבורה, בהשוואה ל-38.4% בלבד עבור אלגוריתמים המתבססים אך ורק על מאפייני TLS.

מאפייני תעבורה במאמר

המאמר משלב שני סוגים של מאפיינים לצורך סיווג התעבורה:

מאפיינים בלתי מוצפנים מתוך ה-TLS handshake, הכוללים:

- גרסת הפרוטוקול של TLS.
- צופן ההצפנה שנבחר.
- אורך ההרחבות בהודעת ה-ClientHello.

מאפיינים מבוססי זרימה (Flow-based Features):

- גודל החבילות.
- זמני הגעה של חבילות (Inter-Packet Time).
- מספר חבילות שנשלחות ומתקבלות במהלך הזרימה.

שיטת סיווג היברידית (Hybrid Classification):

האלגוריתם hRFTC משלב את הנתונים הסטטיסטיים יחד עם מאפייני ה-TLS הזמינים כדי לבצע סיווג מדויק יותר.
הייחודיות של השיטה טמונה בכך שהיא אינה נשענת על SNI, אלא על שילוב של מאפייני זרימה עם נתונים בלתי מוצפנים מה-TLS handshake. בכך היא מאפשרת סיווג גם כאשר נעשה שימוש בהצפנה מלאה, כולל ECH.

תכונות ייחודיות של השיטה:

דיוק גבוה בהשוואה לשיטות קיימות

האלגוריתם hRFTC הגיע לדיוק של 94.6%, לעומת 38.4% בלבד עבור שיטות המבוססות על TLS בלבד.
בהשוואה לשיטות היברידיות אחרות, האלגוריתם צמצם את שיעור הטעויות בכ-50%.

השפעה של משתנים גיאוגרפיים

כאשר האלגוריתם נבחן על נתונים ממדינות שלא נכללו בשלבי האימון, ביצועיו פחתו באופן משמעותי.
מכך עולה הצורך באימון מחדש עבור כל אזור גיאוגרפי, שכן דפוסי התעבורה משתנים בהתאם למיקום.

השפעת הצפנת TLS 1.3 עם ECH

נמצא כי בעוד שהצפנת ECH מסתירה מידע קריטי כמו SNI, חלק מהמאפיינים של TLS נותרים זמינים וניתנים לשימוש לצורך סיווג. עם זאת, שיטות המבוססות על TLS בלבד אינן מספיקות, ולכן יש צורך בשילוב מאפיינים מבוססי זרימה.

יכולת למידה גם עם כמות קטנה של נתוני אימון

האלגוריתם הצליח לשמור על דיוק גבוה גם כאשר אומן על 10% בלבד מהנתונים, והשיג ביצועים דומים לשיטות אחרות שאומנו על 70% מהמאגר.

מסקנות מרכזיות

1. פרוטוקול TLS 1.3 עם ECH מציב אתגר משמעותי בפני סיווג תעבורה בשל הסתרת מידע קריטי כמו SNI.
2. שילוב בין מאפייני TLS למאפיינים סטטיסטיים של זרימה מוביל לשיפור משמעותי בדיוק הסיווג.
3. קיימת תלות משמעותית במיקום גיאוגרפי, ולכן יש צורך באימון מחדש של האלגוריתם עבור אזורים שונים.
4. האלגוריתם hRFTC מפגין יכולת למידה מצוינת, ומסוגל לסווג תעבורה בדיוק גבוה גם עם כמות קטנה יחסית של נתוני אימון.

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

התרומה העיקרית של המאמר

המאמר עוסק ביכולת לזהות את מערכת ההפעלה, הדפדפן והאפליקציה של המשתמש באמצעות תעבורה מוצפנת (HTTPS) בלבד. החוקרים מציגים מתודולוגיה חדשה המסוגלת להבחין בין דפדפנים, מערכות הפעלה ואפליקציות על סמך תכונות רשת ולא תוכן מטען (payload), שאינו זמין בגלל ההצפנה.

<u>תרומות עיקריות:</u>

- **הוכחה ראשונה לכך שאפשר לזהות מערכת הפעלה**, דפדפן ואפליקציה בהתבסס על דפוסי התעבורה בלבד, גם כאשר כל התוכן מוצפן ב-HTTPS.
- **פיתוח והצגה של מאפיינים חדשים לניתוח תעבורה**, המבוססים על התנהגות הבורסטים (bursts) של הדפדפן ופרוטוקול SSL/TLS.
- **יצירת מאגר נתונים רחב** עם יותר מ-20,000 זרימות נתונים מתויגות, הכוללות שילובים של מערכות הפעלה, דפדפנים ואפליקציות שונות.
- **שיפור משמעותי בדיוק הסיווג:** שילוב התכונות החדשות מאפשר זיהוי בדיוק של 96.06% – גבוה משמעותית מהדיוק המושג בשיטות מיון מבוססות התפרצויות.

המאמר מראה שמודלים סטטיסטיים המבוססים על כמות, תזמון, וגודל החבילות בזרימה יכולים לזהות משתמשים ואפליקציות בדיוק גבוה, גם כאשר תוכן התעבורה מוצפן לחלוטין.

מאפייני תעבורה במאמר

<u>מאפיינים בסיסיים:</u>

סטטיסטיקות גודל חבילות:

- מינימום, מקסימום, ממוצע וסטיית תקן של גודל החבילות.

סטטיסטיקות זמני הגעה:

- הפרשי זמן בין חבילות עוקבות (Inter-Arrival Time).
- ממוצע, מקסימום, ומינימום של זמני ההגעה.

ספירת חבילות:

- מספר החבילות שנשלחו קדימה (Forward Packets).
- מספר החבילות שהתקבלו אחורה (Backward Packets).

כמות הנתונים הכוללת:

- סך הבייטים שנשלחו והתקבלו במהלך הזרימה.

וריאציות במדדי התעבורה:

- שונות בגודל החבילות ובזמני ההגעה.

מאפיינים חדשים :

סט חדש של תכונות ייחודיות המבוססות על מבנה פרוטוקול SSL והתנהגות של הדפדפן:

מאפייני SSL/TLS:

- מספר שיטות ההצפנה הנתמכות (Cipher Methods).
- מספר ההרחבות בשלב ה-ClientHello של SSL.
- אורך מזהה הסשן של SSL.
- שימוש בדחיסת SSL Compression Methods.

מאפייני חלונות TCP:

- גודל חלון ההתחלה של TCP Initial Window Size.
- גורם ההגדלה של חלון TCP Window Scaling Factor.

מאפייני קצב שיא (Peak Throughput Features):

- זיהוי "שיאים" בהתנהגות הדפדפן (Bursty Traffic), בהם יש קצב נתונים גבוה לפרקי זמן קצרים.
- מספר הבורסטים שנשלחו וקיבלו.

- קצב המקסימום והממוצע של כל שיא.
- השונות בזמני ההגעה של חבילות השיא.

מדוע המאפיינים החדשים חשובים?

- הם מנצלים חולשות בפרוטוקול SSL כדי להפיק מידע שימושי על הסביבה של המשתמש.
- הם מבוססים על התנהגות אופיינית של דפדפנים, ולכן יכולים להבדיל בין Chrome, Firefox, Safari וכו'.
- הם אינם דורשים גישה למטען התוכן (payload), ולכן רלוונטיים גם כאשר נעשה שימוש בהצפנה מלאה.

ממצאים עיקריים ותובנות

השוואה בין שימוש במאפיינים בסיסיים בלבד לבין שילוב של המאפיינים החדשים:

שיטה	דיוק סיווג
מאפיינים בסיסיים בלבד	93.51%
מאפיינים חדשים	95.03%
שילוב מאפיינים בסיסיים + חדשים	96.06%

שימוש במאפיינים החדשים שיפר את הדיוק בכמעט **2.5%**, והפחית משמעותית את מספר הטעויות.

דיוק הסיווג לפי קטגוריות

- **סיווג מערכת ההפעלה:** 100% דיוק (ללא טעויות כלל).
- **סיווג דפדפן:** 99% דיוק (Safari ו-Chrome היו הקשים ביותר להבחנה).
- **סיווג אפליקציה:** 98% דיוק (טעויות בעיקר בין Facebook ל-Unknown).

ניתוח מטריצות בלבול (Confusion Matrices)

- **מערכות הפעלה:** אין טעויות כלל, הסיווג היה מושלם.
- **דפדפנים:** Safari ו-Chrome היו קשים להבחנה בשל קווי דמיון בחתימת התעבורה.
- **אפליקציות:** טעויות נצפו בעיקר בזיהוי Facebook, אשר סווגה כ-"Unknown" בכ-29% מהמקרים.

זיהוי "בורסטים" בתעבורה שיפר את הביצועים

לכל דפדפן יש אופי שונה של זרימות נתונים, כך שהמאפיינים החדשים אפשרו הבחנה מדויקת יותר.
זוהו "שיאים" אופייניים עבור YouTube, Twitter, Facebook, מה שאיפשר סיווג אפליקציות בדיוק גבוה.

מסקנות מרכזיות

1. ניתן לסווג מערכת הפעלה, דפדפן ואפליקציה בדיוק גבוה, למרות שהתעבורה מוצפנת ב-HTTPS.
2. הוספת תכונות מבוססות SSL והתנהגות דפדפנים משפרת משמעותית את הביצועים.
3. שימוש במידע זה יכול לשמש הן לגורמים לגיטימיים (כגון ספקי שירותי רשת) והן לגורמים עוינים (כגון האקרים).
4. בעתיד ניתן יהיה לזהות לא רק את הסביבה של המשתמש, אלא גם את הפעולות שהוא מבצע (כגון שליחת ציוץ או צפייה בסרטון).

חלק ג

גרסת הפייתון שבה השתמשנו כדי להריץ את כלל הקודים:

```
$ python3  - -version

Python 3.11.9
```

גרף A

הגרף מציג את התפלגות הפרוטוקולים הנפוצים ביותר בכל אחת מחמשת האפליקציות.

ציר ה-X:
מציג את שמות הפרוטוקולים שנמצאו בתעבורת הרשת של כל אחת מן האפליקציות הנבדקות
הפרוטוקולים מוצגים לפ תדירות השימוש שלהם בכל אחת מהאפליקציות

ציר ה-Y:

מייצג את התדירות שכל פרוטוקול מופיע בתעבורת הרשת של באפליקציה. ככל שהערך גבוה יותר כך הפרוטוקול היה נפוץ יותר במהלך השימוש באפליקציה

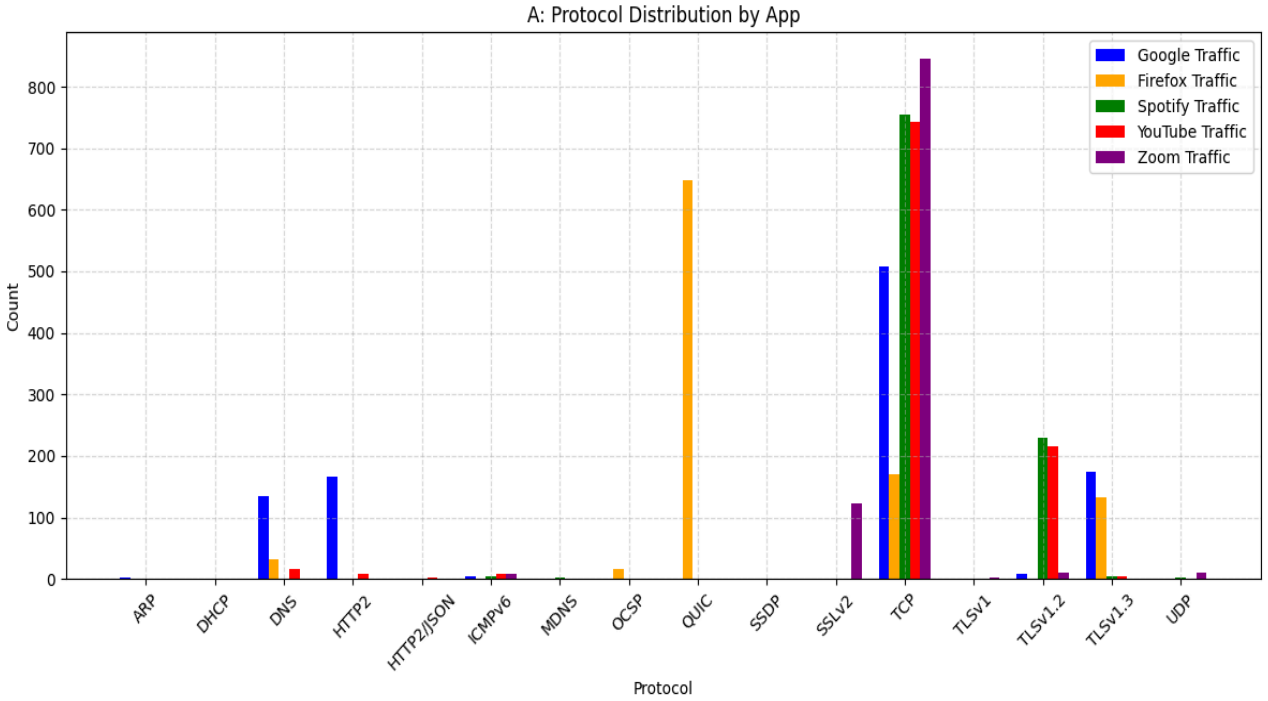
מה ניתן ללמוד מהגרף ?

ניתוח דפוסי השימוש בפרוטוקולים

הגרף מאפשר לזהות את הפרוטוקולים הדומיננטיים בכל אפליקציה.תדירות הופעת הפרוטוקולים מרמזת על סוגי התעבורה שבהם משתמש השירות.

השוואה בין האפליקציות והשירותים השונים

- **Google** מציגה שימוש ניכר בפרוטוקולי DNS ו-HTTP, שירותי Google תלויים רבות בביצועי שאילתות DNS מהירים ובבקשות HTTP.
- **Firefox** משתמש הרבה בפרוטוקול QUIC, מה שמעיד על שימוש בפרוטוקול זה כדי לשפר את מהירות טעינת הדפים והפחתת השהיות.
- **YouTube ו-Spotify** מציגים שימוש משמעותי ב-TCP ו-TLS, מה שמדגיש את הצורך בהעברת נתונים מאובטחת ואמינה.
- **Zoom** מסתמך בעיקר על TCP ו-SSL, מה שמעיד על שימוש בפרוטוקולים מבוססי הצפנה לתקשורת שלו.



הגרף מציג את התפלגות הדגלים הנפוצים ביותר בכל אחת מחמשת האפליקציות.

דגלים נפוצים

- SYN (0x2)– משמש לפתיחת חיבור TCP.

- ACK (0x10) - משמש לאישור קבלת נתונים
- FIN (0x1) - מסמן סיום חיבור
- PSH (0x8) - לשלוח או לקבל את הנתונים באופן מיידי
- (0x0) - חבילה ללא דגלים פעילים

הגרף מציג את תדירות הדגלים ששימשו בפרוטוקולים של חמשת השירותים:

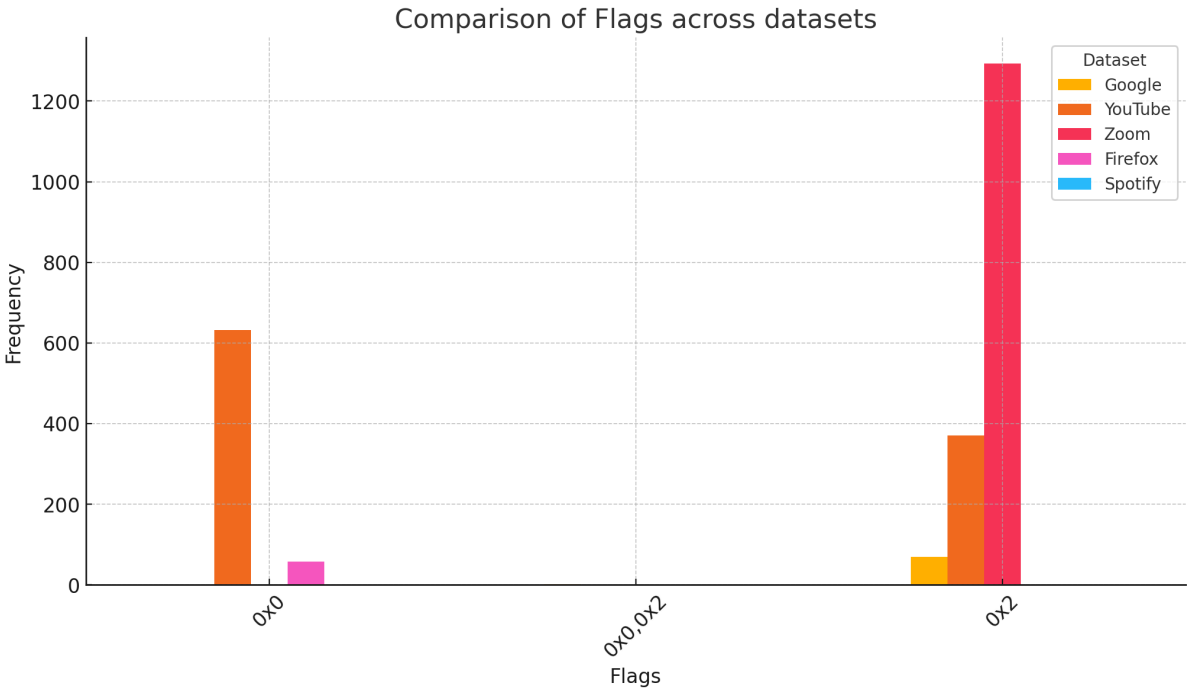
ציר ה-X: הערכים של הדגלים כאשר.

ציר ה-Y: מספר הפעמים שכל ערך הופיע בתעבורה של כל אפליקציה.

מה ניתן ללמוד מהגרף?

השוואה בין אפליקציות שונות

- Zoom מציג שימוש גבוה בדגל 0x2, המעיד על יצירת חיבורים חדשים בתדירות גבוהה, דבר התואם את אופי השימוש בשיחות וידאו.
- YouTube מציג שימוש משולב בין 0x2 ו 0x0, מה שמעיד על שילוב בין חיבורים חדשים לתעבורה רציפה.
- Spotify, Firefox ו-Google מציגים שכיחות נמוכה יותר של דגלים אלה, דבר שעשוי להעיד על שימוש בחיבורים יציבים וארוכי טווח במקום יצירת חיבורים חדשים באופן תדיר.



התפלגות לפי TTL

TTL הוא ערך מספרי בשדה כותרת של חבילה בפרוטוקול IP, המשמש להגבלת חייה של החבילה ברשת. בכל פעם שהחבילה עוברת דרך נתב, הערך שלה מופחת באחד. כאשר TTL מגיע ל-0, החבילה מושלכת. מטרתו למנוע לולאות נצחיות ברשת ולאפשר ניהול טוב יותר של תעבורת הנתונים.

הגרף מציג את ההתפלגות של ערכי TTL בתעבורת הרשת של האפליקציות השונות.

- **ציר X:** מציג את ערכי ה-TTL כפי שנמצאו בחבילות הרשת.
- **ציר Y:** מציג את התדירות שבה כל ערך TTL הופיע בתעבורה.

מה ניתן ללמוד מהגרף?

זיהוי יעדי תקשורת נפוצים:

ניתן לראות שערכים מסוימים של TTL מופיעים בתדירות גבוהה יותר, מה שעשוי להצביע על השרתים המרוחקים שנמצאים בשימוש תדיר.

השוואה בין אפליקציות שונות:

- Zoom מציג TTL נמוך יותר, מה שעשוי להצביע על כך שהשרתים שהוא מתקשר איתם קרובים יותר גיאוגרפית למשתמשים.
- Zoom מציג TTL נמוך יותר, מה שעשוי להצביע על כך שהשרתים שהוא מתקשר איתם קרובים יותר גיאוגרפית למשתמשים.
- YouTube מציג TTL גבוה יותר, ככל הנראה עקב פנייה לשרתי CDN מרוחקים.
- Spotify ו-Firefox מציגים פיזור TTL רחב יותר, מה שעשוי להצביע על ניתובים שונים כתלות בתוכן המבוקש ובמיקום השרתים.

דפוסי אבטחה וניתוח תקשורת:

- חבילות עם TTL נמוך מאוד עשויות להעיד על תעבורה מקומית או על שירותים שמתארחים קרוב למשתמשים
- TTL גבוה עשוי להופיע כאשר תעבורה עוברת דרך מספר רב של נתבים, דבר שעשוי לאותת על שימוש בשירותים חיצוניים מרוחקים.

גרף B

מה מייצג הגרף ?

הגרף מייצג את **התפלגות כתובות היעד** הנפוצות ביותר בכל אחת מ-5 האפליקציות מסוגים שונים.

ציר ה - X:

- מציג את כתובות היעד (Destination Address) או הפורטים שאליהם נשלחו הנתונים.
- לכל אפליקציה יוצגו חמש הכתובות/הפורטים הנפוצים ביותר.
- הכתובות עשויות להיות מספרי פורטים או כתובות IP ספציפיות.
- הכתובות מסודרות לפי כמות ההופעות שלהן.

<u>ציר ה - Y:</u>

- מייצג את מספר הפעמים שכל כתובת יעד הופיעה בתעבורת הרשת של אותה אפליקציה.

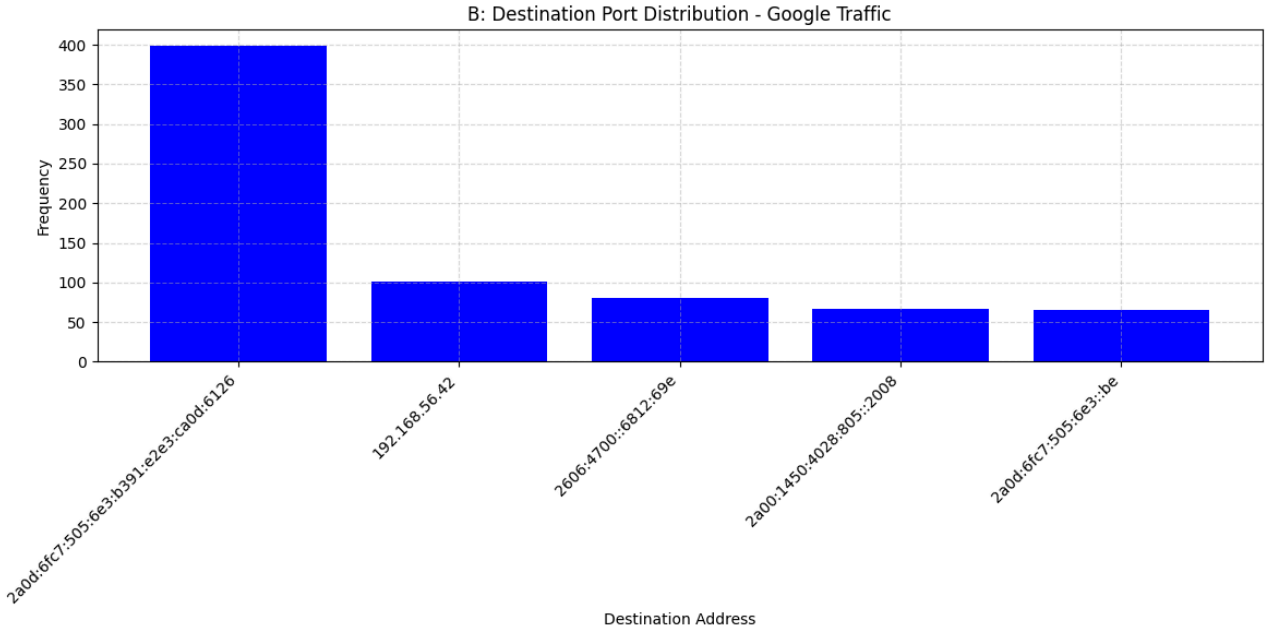
<u>מה ניתן ללמוד מהגרף ?</u>

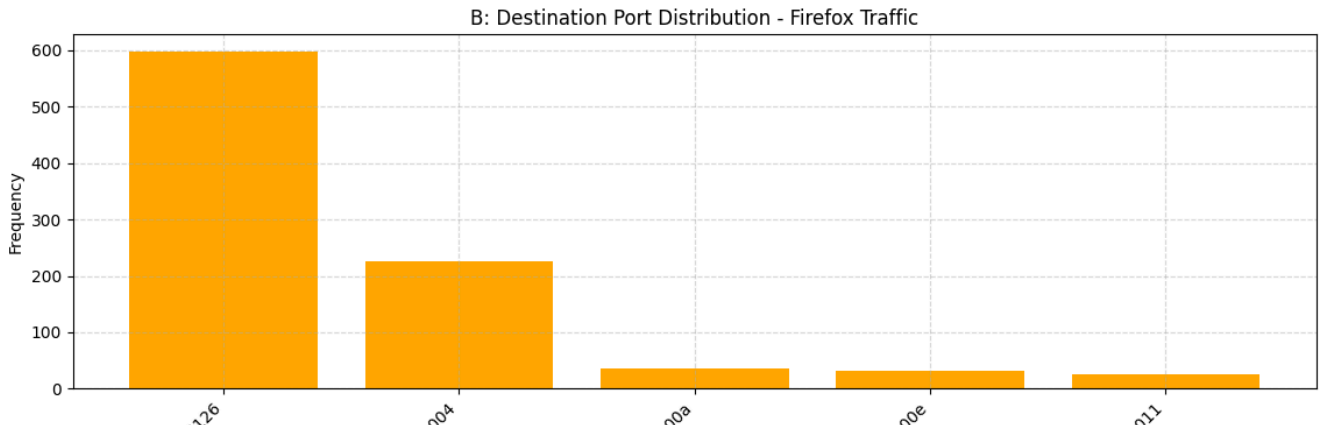
זיהוי יעדי תקשורת נפוצים:

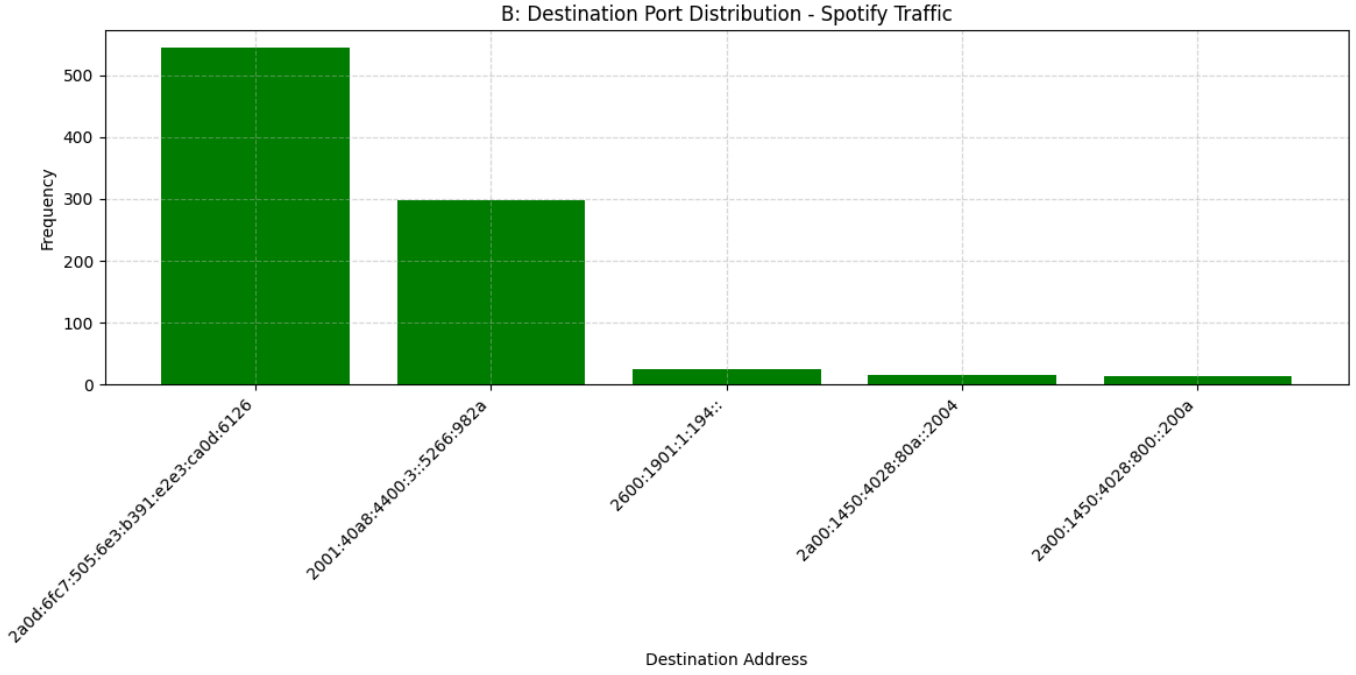
- ניתן לראות אילו כתובות יעד או פורטים היו בשימוש הנפוץ ביותר לכל אפליקציה.

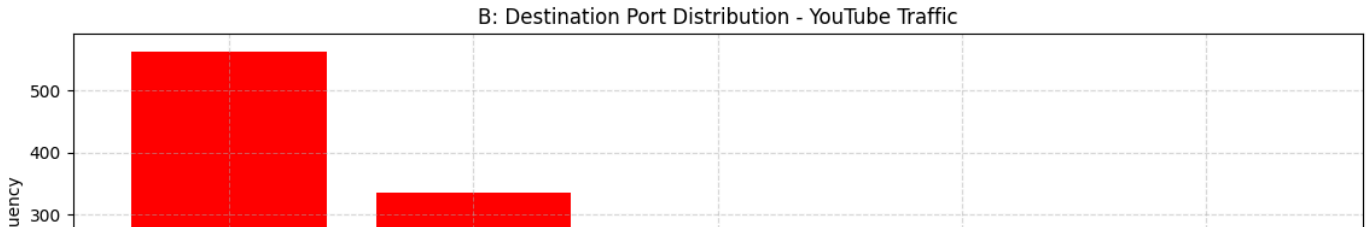
השוואה בין אפליקציות שונות:

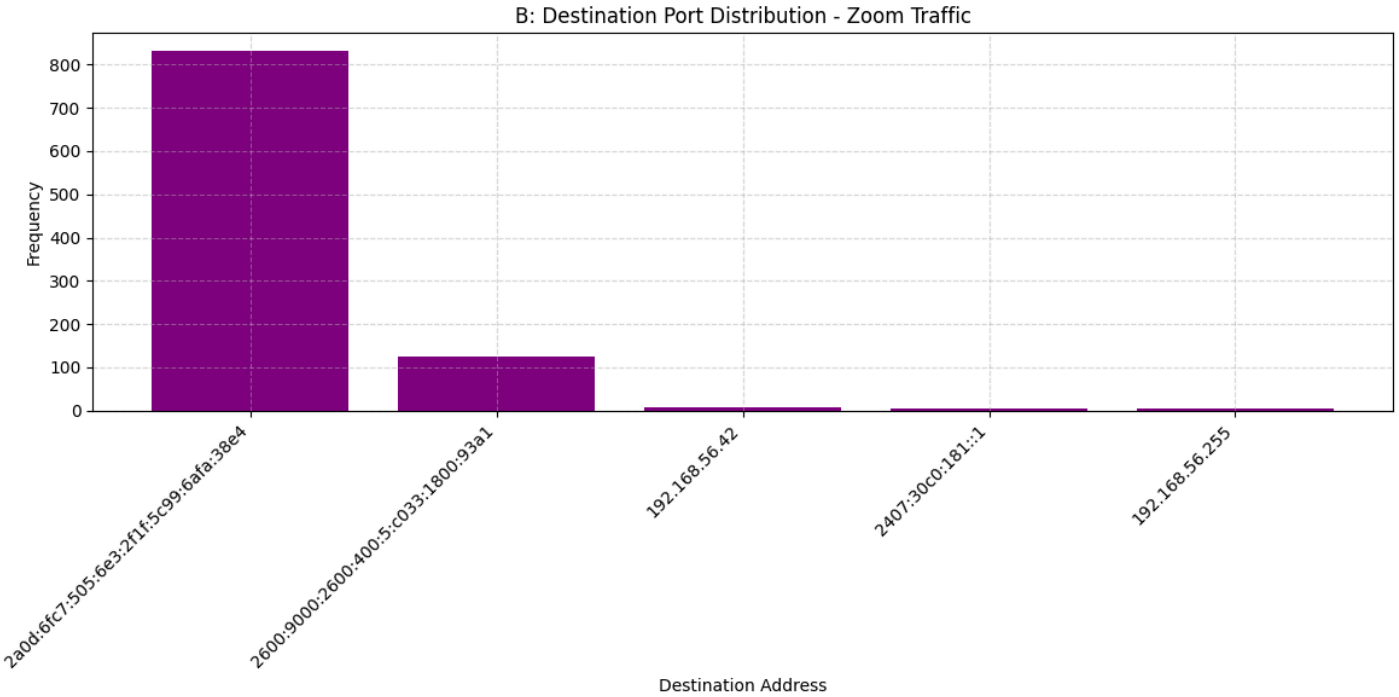
- ניתן לראות הבדלים בין כתובות היעד של אפליקציות שונות.
- למשל, Zoom עשוי להציג כתובות של שרתי וידאו בזמן אמת.






B: Destination Port Distribution - Spotify Traffic
Frequency
0
100
200
300
400
500
2a0d:6fc7:505:6e3:b391:e2e3:ca0d:6126
2001:40a8:4400:3::5266:982a
2600:1901:1:194::
2a00:1450:4028:80a::2004
2a00:1450:4028:800::200a
Destination Address


B: Destination Port Distribution - YouTube Traffic
500
400
300


B: Destination Port Distribution - Zoom Traffic
Frequency
800
700
600
500
400
300
200
100
0
2a0d:6fc7:505:6e3:2f1f:5c99:6afa:38e4
2600:9000:2600:400:5:c033:1800:93a1
192.168.56.42
2407:30c0:181::1
192.168.56.255
Destination Address

גרף C

מה מייצג הגרף ?

הגרף מייצג את סוגי ה - Handshake הנפוצים ביותר בפרוטוקול TLS לכל אחת מ - 5 האפליקציות מסוגים שונים.

ציר ה - X:

- מציג את סוגי ה - Handshake

שני הסוגים העיקריים:

1. Client Hello אשר הלקוח מתחיל תקשורת מאובטחת עם השרת.
2. Server Hello כאשר השרת משיב ללקוח ומאשר את הפרמטרים ליצירת החיבור המאובטח

ציר ה - Y:

- מציג את **מספר הפעמים** שכל סוג של Handshake הופיע בתעבורת הרשת של האפליקציה.
-

מה ניתן ללמוד מהגרף ?

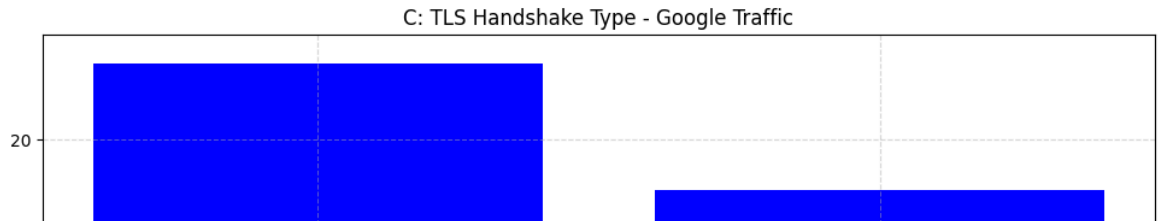
- ניתן לראות איזו אפליקציה יוצרת הכי הרבה חיבורים TLS חדשים.

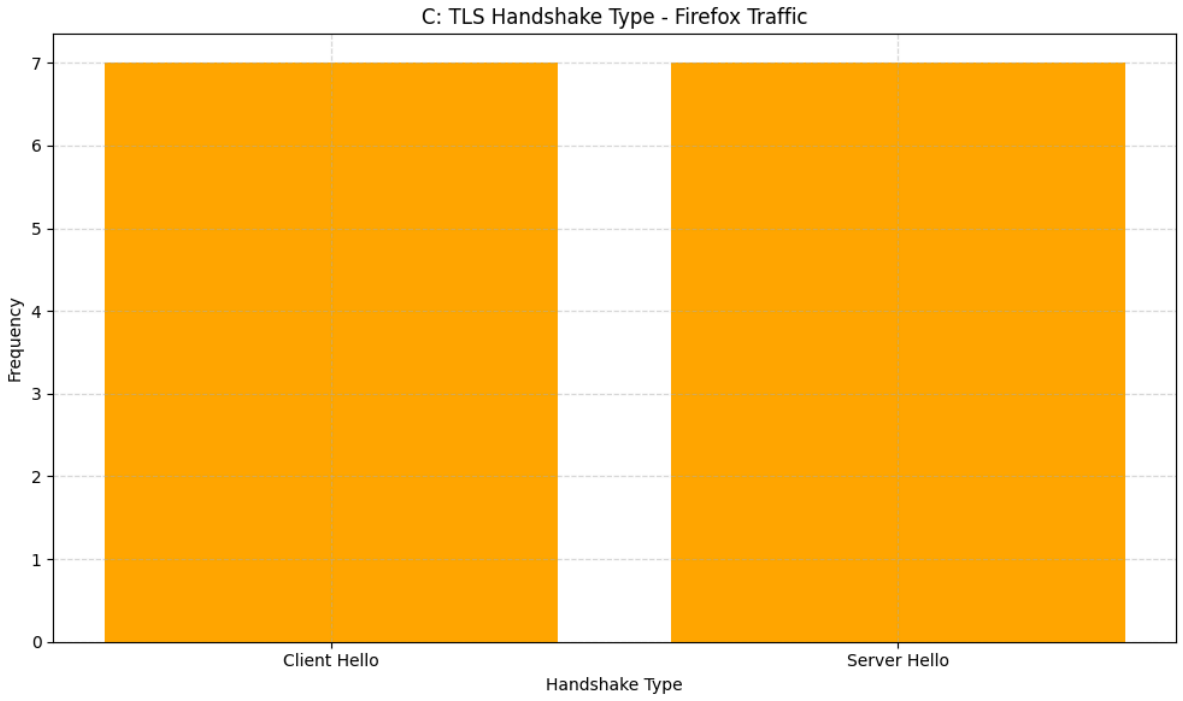
השוואה בין אפליקציות שונות:

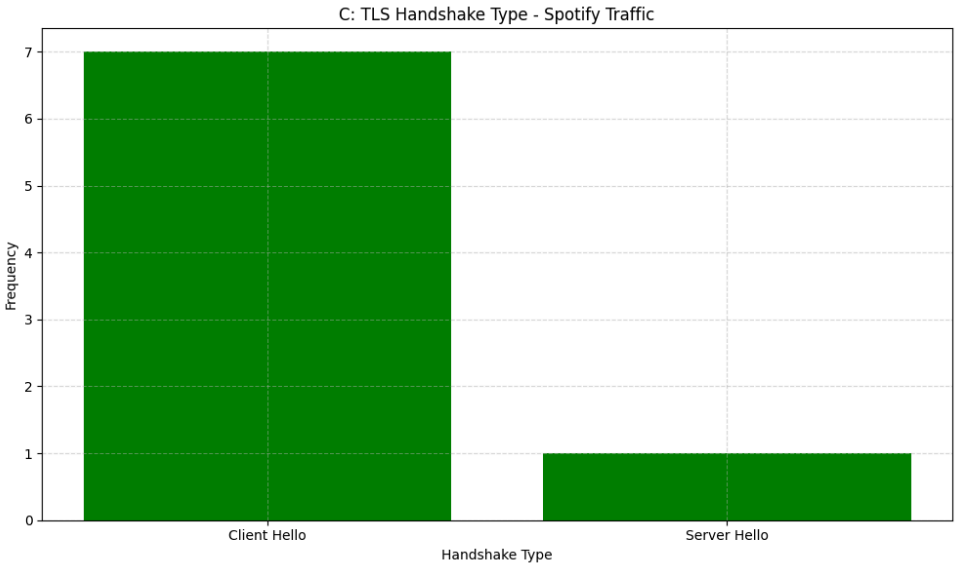
- אפליקציות כמו דפדפנים **(Google, Firefox)** מציגות הרבה Client Hello, כי הן פותחות חיבורים מאובטחים רבים.
- **Zoom** מציג הרבה Client Hello, ייתכן שהאפליקציה פותחת חיבורים נפרדים לכל משתמש או לכל שיחת וידאו.
- **Spotify** מציג פחות Server Hello, ייתכן שהוא מפחית את מספר ה-"לחיצות יד" על ידי מחזור חיבורים קיימים או שימוש בפרוטוקול QUIC.

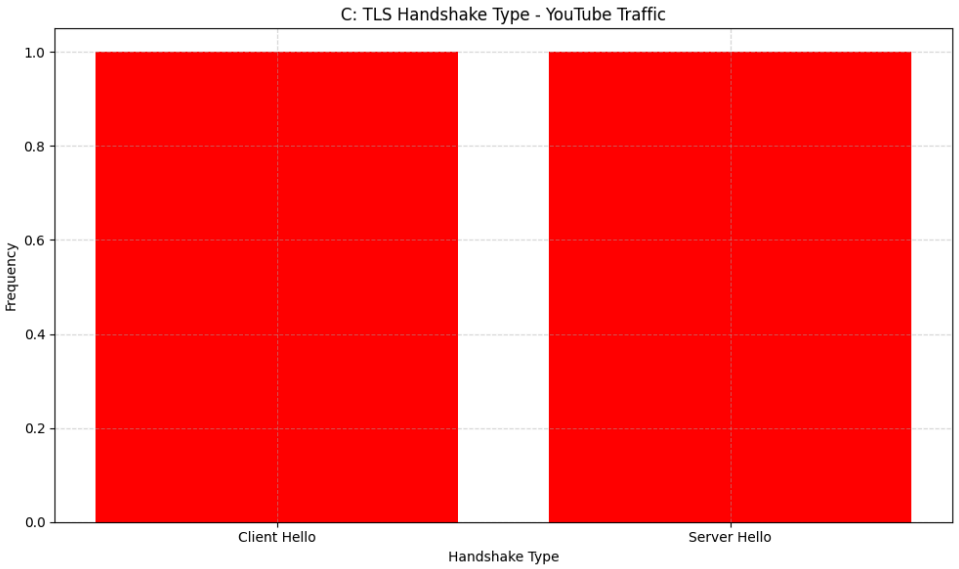
זיהוי בעיות ברשת:

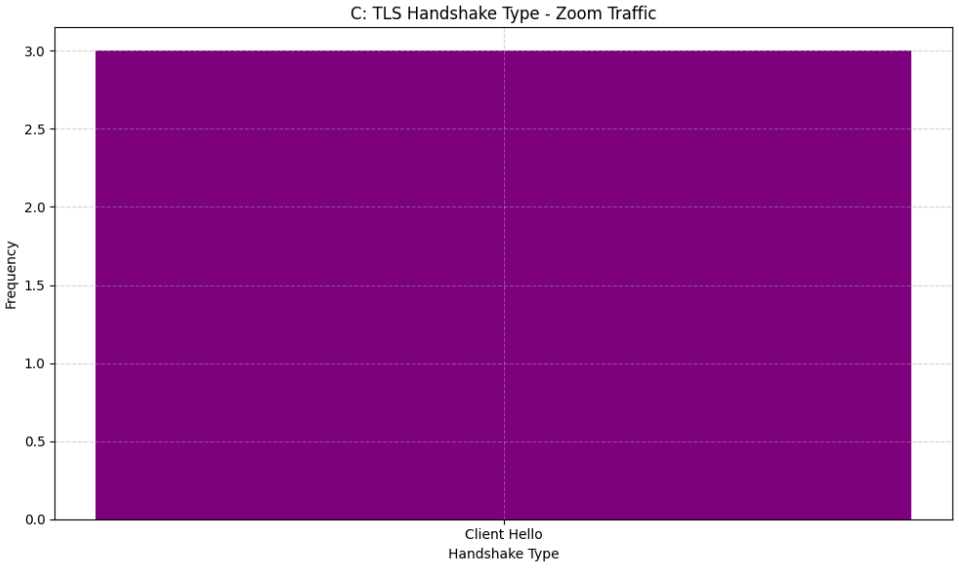
- הרבה מאוד Handshake בפרק זמן קצר עשוי להצביע על בעיה בתקשורת, כמו ניתוקים תכופים או ניסיון של אפליקציה ליצור יותר חיבורים מהנדרש.
- אם רואים רק Client Hello וללא Server Hello, זה עשוי להעיד על תקלה בגישה לשרתים או חסימה של חיבורים מאובטחים.




C: TLS Handshake Type - Firefox Traffic
Frequency
0
1
2
3
4
5
6
7
Client Hello
Server Hello
Handshake Type


C: TLS Handshake Type - Spotify Traffic
Frequency
0
1
2
3
4
5
6
7
Client Hello
Server Hello
Handshake Type


C: TLS Handshake Type - YouTube Traffic
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
Client Hello
Server Hello
Handshake Type


C: TLS Handshake Type - Zoom Traffic
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Client Hello
Handshake Type

גרף D

מה מייצג הגרף ?

הגרף מייצג את **התפלגות גדלי החבילות** שנשלחו ונקלטו על ידי כל אחת מ-5 האפליקציות מסוגים שונים.

ציר ה - X:

- מציג את גודל החבילה (Packet Size) בבייטים (Bytes).
- ציר זה מחולק לטווחים, למשל, חבילות בטווחים של 0-50, 50-100, 100-150 בייטים וכו'.
- החבילות מחולקות לקטגוריות כך שניתן לראות באילו טווחים מופיעות הכי הרבה חבילות.

ציר ה - Y:

- מייצג את מספר הפעמים שכל גודל חבילה הופיע בתעבורת האפליקציה (כלומר, כמות החבילות בגודל מסוים).

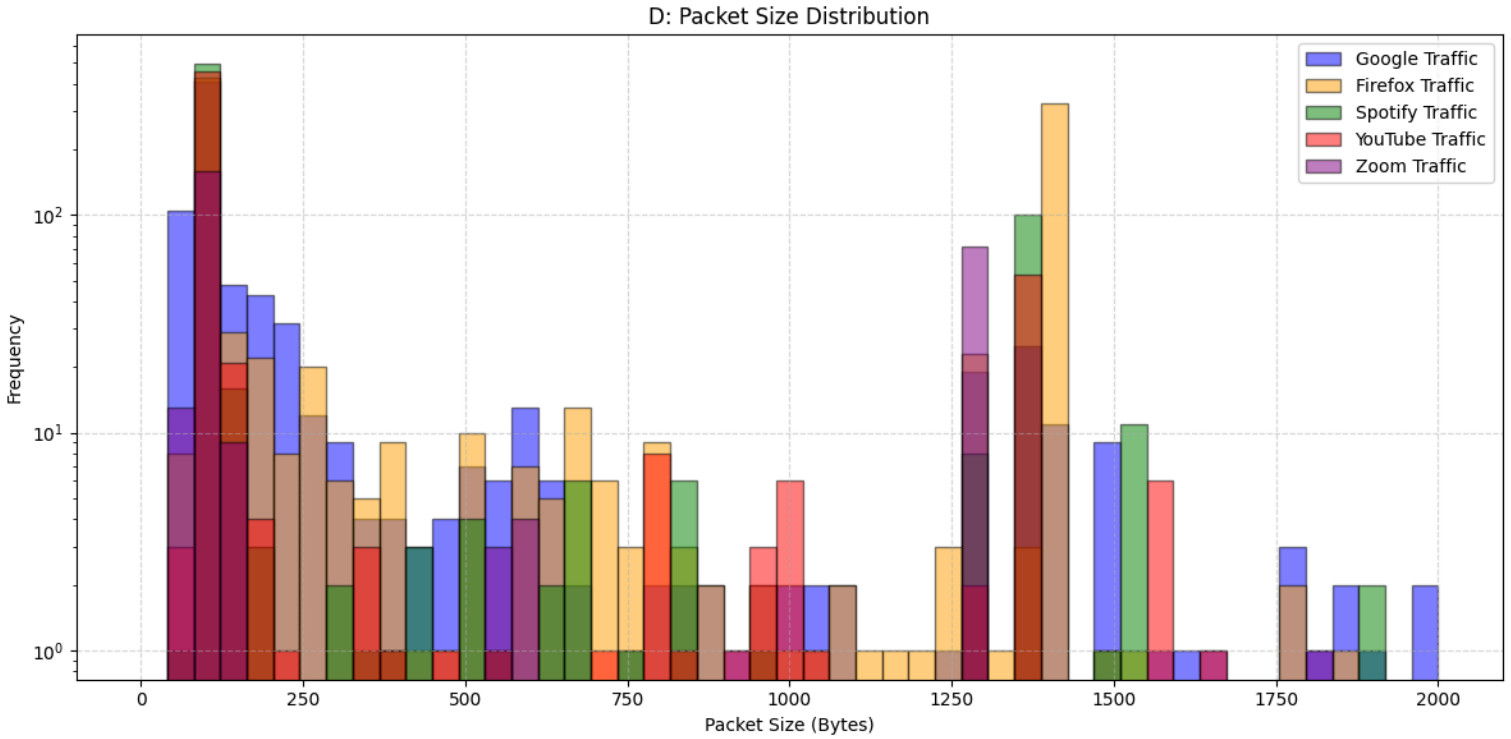
מה ניתן ללמוד מהגרף ?

השוואת דפוסי תעבורה בין אפליקציות שונות

- **דפדפנים (Google, Firefox)** מציגים הרבה חבילות קטנות בגלל בקשות HTTP קצרות.
- **YouTube, Zoom ו-Spotify** מציגים הרבה חבילות גדולות (קרובות ל-1500 בייטים), כי הם שולחים מדיה זורמת.

זיהוי בעיות ברשת

- אם יש הרבה חבילות קטנות מאוד, זה עלול להצביע על חיבור לא יציב.
- אם אין כמעט חבילות בגודל 1500 בייטים, ייתכן שהרשת משתמשת במקטעים קטנים יותר, מה שעלול לגרום ליעילות נמוכה יותר.



גרף E

מה מייצג הגרף ?

הגרף מייצג את **התפלגות זמני ההגעה** בין חבילה אחת לשניה על ידי כל אחת מ-5 האפליקציות מסוגים שונים.

ציר ה - X:

- מייצג את הפרשי הזמנים (Inter-Arrival Time) בשניות בין שתי חבילות עוקבות שנקלטו באפליקציה.
- הציר מחולק למרווחי זמן, למשל: 0-0.01 שניות, 0.01-0.02 שניות וכו'.

ציר ה - Y:

- מציג את כמות החבילות שהתאימו לכל טווח זמן בין הגעתן.

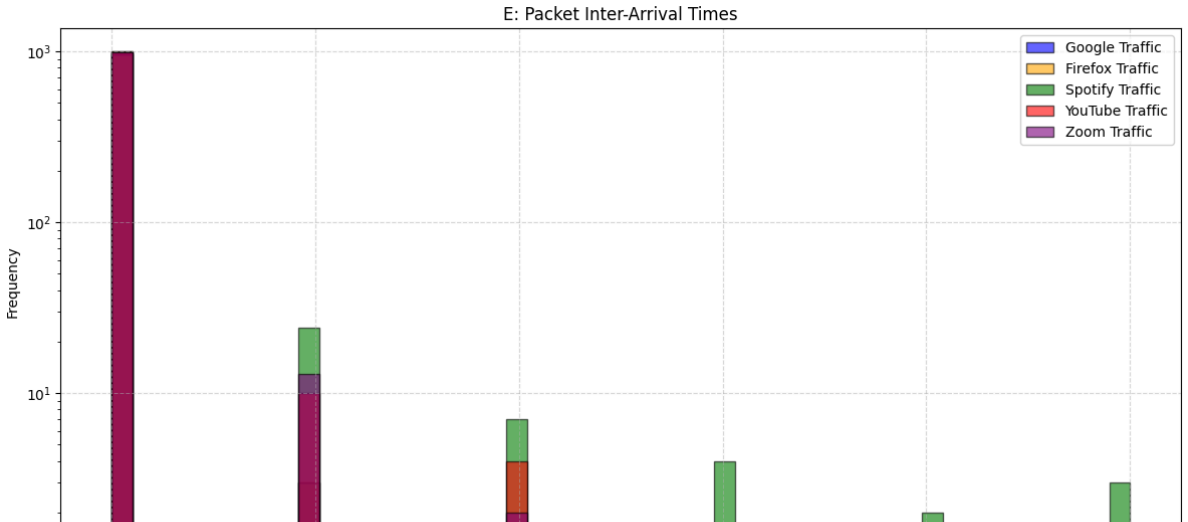
מה ניתן ללמוד מהגרף ?

ניתוח דפוסי תעבורה ברשת

- אם רוב החבילות נמצאות בטווחי זמן קטנים (למשל, < 0.01 שניות) → סימן שהאפליקציה שולחת נתונים באופן רציף וחלק (כגון סטרימינג).
- אם יש קפיצות או מרווחי זמן גדולים בין חבילות → סימן שהאפליקציה שולחת נתונים באופן מקוטע (כגון גלישה באתרים).

השוואה בין אפליקציות שונות

- **YouTube ו-Zoom** בעלי זמני הגעה קטנים יותר, נראה כי רוב החבילות שלהם מגיעות בהפרשים קטנים מאוד, מה שמעיד על כך שהפרוטוקולים שלהם מותאמים להזרמת נתונים בקצב גבוה ובאופן רציף.
- **Spotify** מציג זמני הגעה גדולים יותר, חבילות של Spotify מתפלגות באופן רחב יותר, דבר שעשוי להעיד על מנגנוני אגירה (buffer) בתעבורת סטרימינג של מוזיקה.
- **Firefox, Google** עשויות להראות זמני הגעה מגוונים יותר, כי תעבורת דפדפן משתנה לפי דפוסי הגלישה.



גרף F

מה מייצג הגרף ?

הגרף מייצג את **מספר החבילות (packets)** שנשלחו או התקבלו עבור כל אחת מ-5 האפליקציות מסוגים שונים.

ציר ה - X:

- מייצג את האפליקציות שנבדקו

ציר ה - Y:

- מציג את מספר החבילות (Packets Count) שנקלטו על ידי כל אפליקציה.

מה ניתן ללמוד מהגרף ?

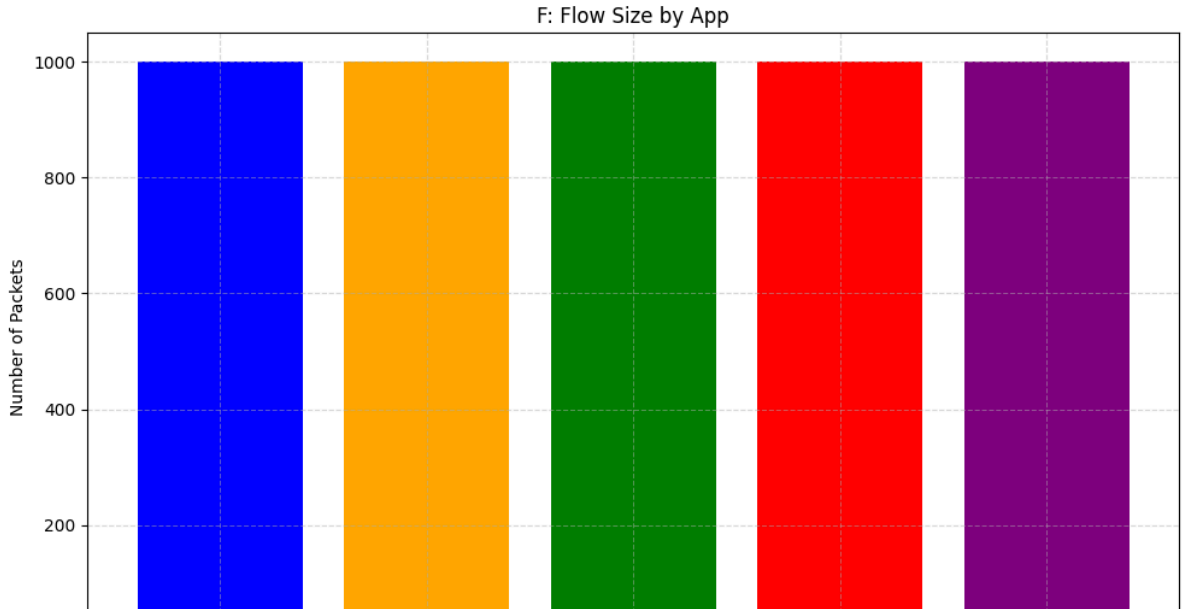
ניתוח היקף התקשורת של כל אפליקציה

- אפליקציה עם עמודה גבוהה משתמשת ביותר חבילות, כלומר שולחת ומקבלת הרבה נתונים.
- אפליקציה עם עמודה נמוכה משתמשת בפחות חבילות, כלומר יש לה פחות תעבורה ברשת.

השוואת צריכת התעבורה בין אפליקציות שונות

- **YouTube ו-Zoom** צפויים להציג מספר גבוה של חבילות, כיוון ששירותי סטרימינג ושיחות וידאו משתמשים בזרם קבוע של נתונים.
- **Google ו-Firefox** עשויים להציג פחות חבילות יחסית, כיוון שתעבורת דפדפן מתרחשת בקפיצות.
- **Spotify** עשוי להציג כמות בינונית של חבילות, כי מוזיקה נשלחת במקטעים קטנים ולא זורמת כל הזמן כמו וידאו.

הערה: בפועל הגרף לא מראה את ההבדל בין כמות החבילות כי לקחנו מדגמית 1000 חבילות מכל סוג תעבורה כדי ששאר הנתונים יהיו אובייקטיבים.



גרף G

מה מייצג הגרף ?

הגרף מייצג את **סך כל הנתונים (Total Bytes Transmitted)** שנשלחו או התקבלו עבור כל אחת מ-5 האפליקציות מסוגים שונים.

ציר ה - X:

- מציג את האפליקציות שנבדקו.

ציר ה - Y:

- מציג את סך כל הנתונים (Total Bytes Transmitted) בכל האפליקציה, כלומר כמות המידע שהועברה (נשלחה או התקבלה) בפועל.

מה ניתן ללמוד מהגרף ?

ניתוח נפח התעבורה לפי אפליקציה

- אפליקציה עם עמודה גבוהה שולחת ומקבלת הרבה מאוד נתונים.
- אפליקציה עם עמודה נמוכה משתמשת בפחות נפח נתונים, מה שמעיד על שימוש מועט או אפליקציה קלה יותר מבחינת תעבורה.

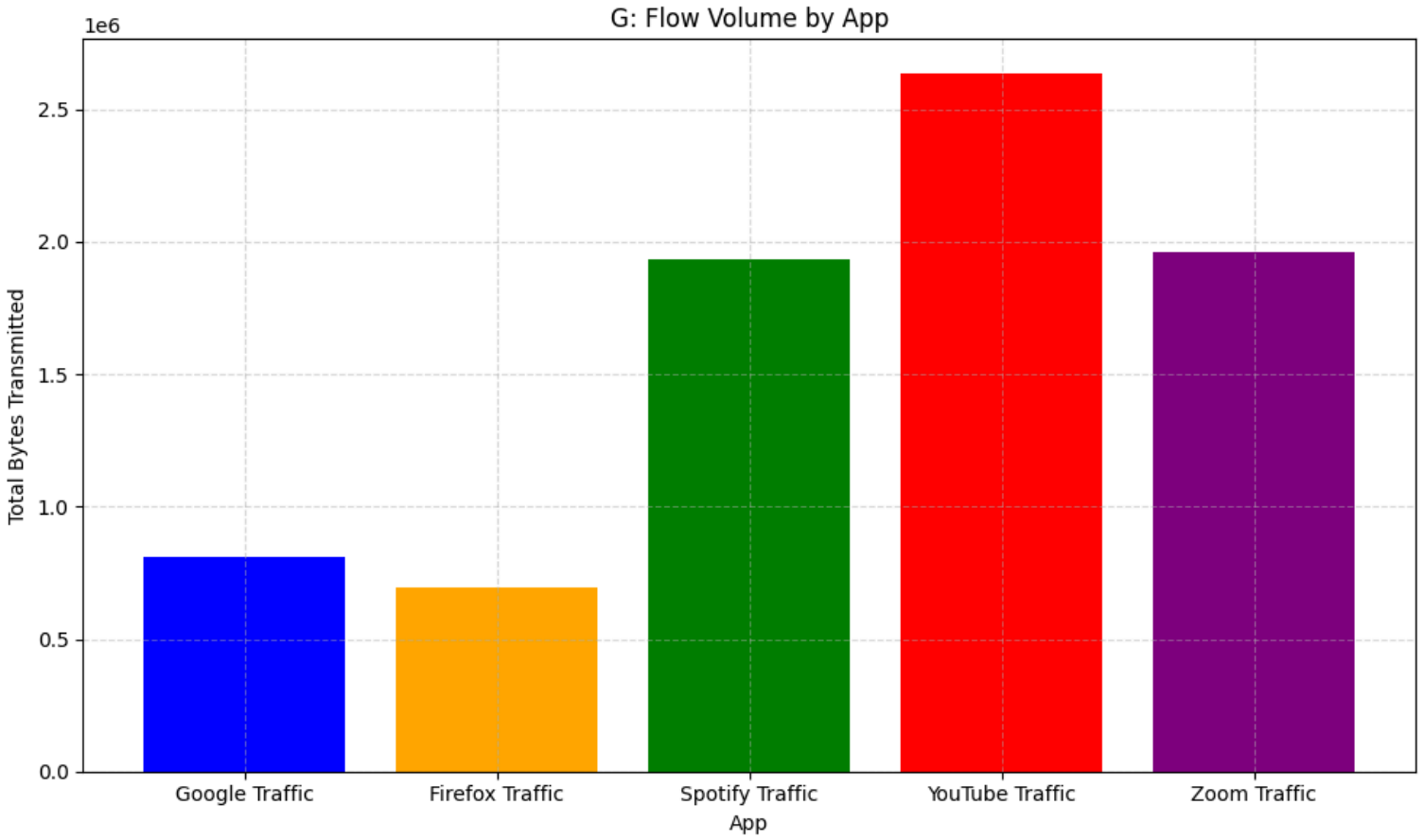
השוואת נפח נתונים בין אפליקציות שונות

- **YouTube** מייצר את נפח הנתונים הגדול ביותר, מפני שהוא שירות מבוסס וידאו, והזרמת וידאו דורשת כמות גדולה של נתונים.
- **Zoom** גם מעביר כמות גדולה של נתונים, מה שמעיד על כך ששיחות וידאו בזמן אמת צורכות הרבה רוחב פס, במיוחד עבור וידאו באיכות גבוהה.
- **Spotify** צורך פחות מנתוני YouTube ו-Zoom אבל יותר מדפדפנים. סטרימינג של מוזיקה דורש פחות נתונים מווידאו, אך עדיין יותר מדפדפנים.
- **Google** ו-**Firefox** בעלי נפח נתונים נמוך יחסית, דפדפנים מסתמכים יותר על טעינת דפים ותגובות HTTP קצרות, מה שגורם לצריכת נתונים נמוכה יותר בהשוואה לשירותי מדיה.

הבדלים בין מספר החבילות לנפח הנתונים

- אם אפליקציה מסוימת שלחה המון חבילות אבל עם נפח נתונים קטן, זה מצביע על כך שהיא משתמשת בהרבה חבילות קטנות (למשל, תקשורת HTTP עם דפים קטנים).
- אם אפליקציה שלחה מעט חבילות עם נפח נתונים גבוה, זה מצביע על כך שהיא שולחת חבילות גדולות יותר, כנראה מדיה כבדה.

אופק בר שלום - 324161421 ג'סי בריגס - 314708041 עמית דנינו - 323084434 רועי נאור - 323917104



איסוף נתונים וחלוקה

מאגר הנתונים כולל רשומות תעבורה המסומנות בהתאם ליישומים שונים. לצורך אימון והערכת המודל, השתמשנו בחלוקה של **80-20**, כאשר **80% מהנתונים שימשו לאימון** ו-**20% לבדיקות**. שיטה זו מבטיחה למידה יעילה תוך שמירה על קבוצת בדיקה בלתי תלויה להערכת הביצועים.

מדוע Random Forest

בחרנו באלגוריתם **Random Forest** בשל עמידותו בטיפול בנתונים מובנים ויכולתו לזהות גבולות החלטה מורכבים. האלגוריתם מבוסס על חיזוי באמצעות מספר רב של עצי החלטה, מה שמשפר את הדיוק ומקטין את הסיכון להתאמת יתר (Overfitting). יתרונותיו המרכזיים כוללים:

- **התמודדות עם נתונים מרובי ממדים**, מה שהופך אותו למתאים במיוחד לסיווג תעבורת רשת.
- **עמידות להתאמת יתר** באמצעות שילוב של מספר עצים להחלטה.
- **יכולת פרשנות**, שכן ניתן לזהות אילו משתנים הם החשובים ביותר בתהליך הסיווג.

3. תרחישי תקיפה וניתוח נתונים

אנו מנתחים את ביצועי המודל בשני מצבי תקיפה:

1. **תרחיש 1:** התוקף יודע את גודל המנות, חותמות הזמן, וזיהוי הזרימה המקודד.
2. **תרחיש 2:** התוקף יודע רק את גודל המנות וחותמות הזמן.

3.1 מדדי הערכה

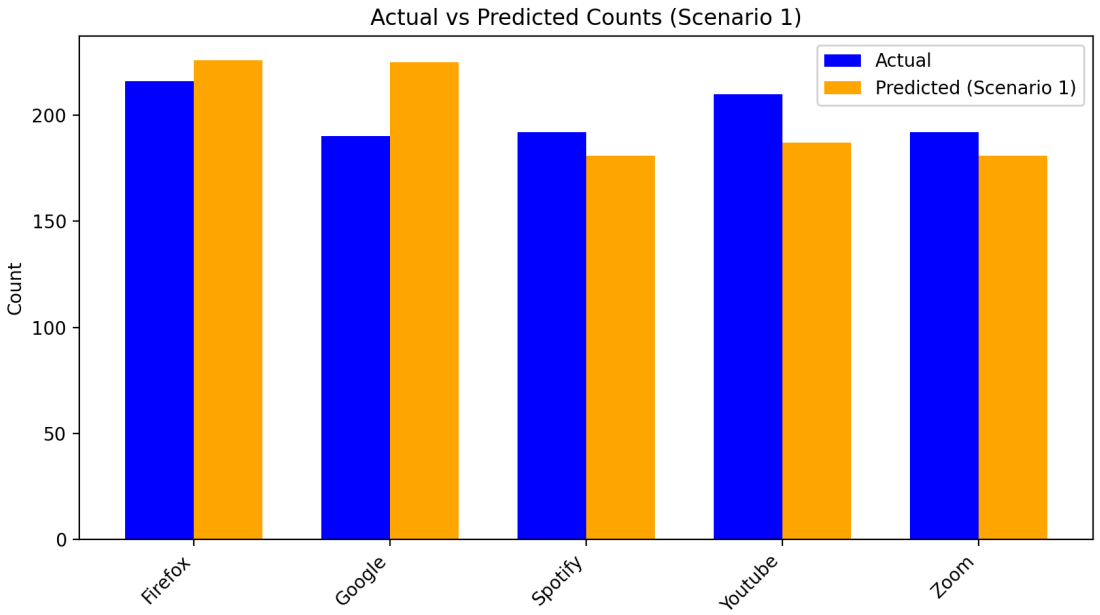
כדי להעריך את ביצועי המודל, אנו משתמשים במדדים הבאים:

- **Precision (דיוק):** מדד המייצג כמה מהתחזיות החיוביות אכן נכונות.
- **Recall (זיהוי):** מדד הבודק כמה מהמקרים בפועל זוהו כראוי.
- **F1-Score:** מדד המשלב דיוק וזיהוי להערכת האמינות הכוללת של המודל.
- **Support (תמיכה):** מספר הדוגמאות השייכות לכל קטגוריה בסט הנתונים.

תוצאות ופרשנות

תרחיש 1 (עם נתוני תוצאות ופרשנות)

אחוז דיוק: 91.1%



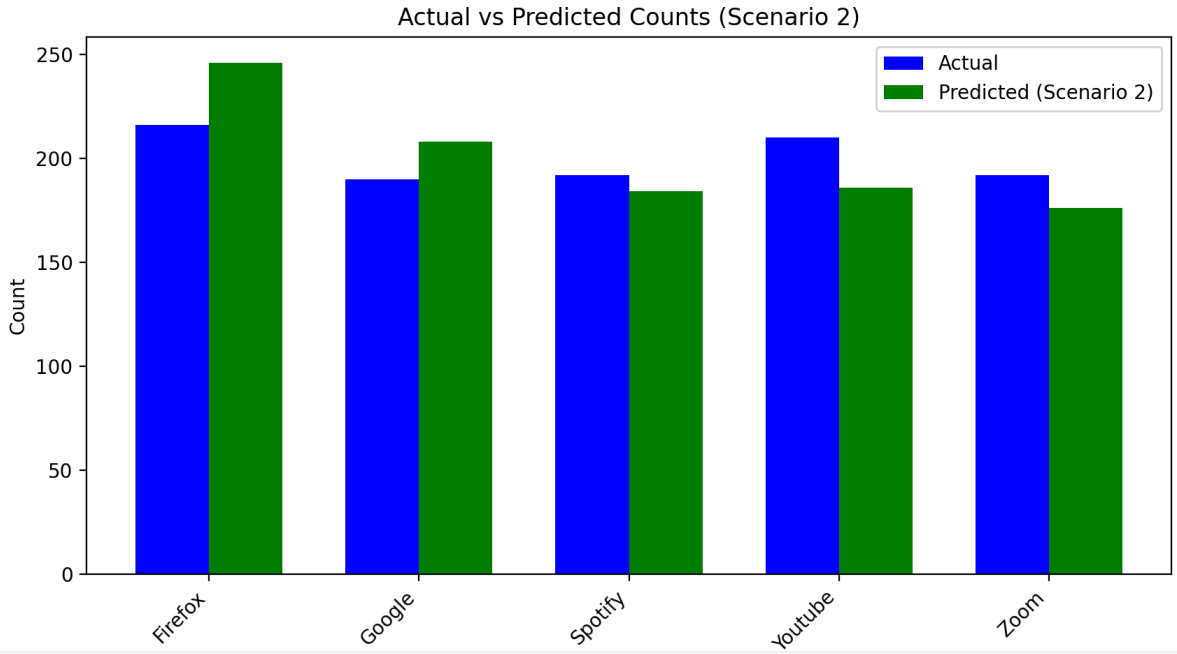
Application	Precision	Recall	F1-Score	Support
Firefox	88%	92%	90%	216
Google	82%	97%	89%	190
Spotify	94%	89%	91%	192
YouTube	96%	85%	90%	210

Zoom	**99%**	**93%**	**96%**	**192**

Application	**Actual**	**predicted**
Firefox	**22%**	**23%**
Google	**19%**	**22%**
Spotify	**19%**	**18%**
Youtube	**21%**	**19%**
Zoom	**19%**	**18%**

תרחיש 2 (ללא נתוני זרימה)

אחוז דיוק: 87.6%



Application	Precision	Recall	F1-Score	Support
Firefox	78%	88%	83%	216
Google	77%	85%	81%	190
Spotify	91%	88%	89%	192
Youtube	97%	86%	91%	210
Zoom	100%	92%	96%	192

Application	Actual	Predicted
Firefox	22%	25%

Google	**19%**	**21%**
Spotify	**19%**	**18%**
Youtube	**21%**	**19%**
Zoom	**19%**	**18%**

ניתוח תוצאות

תרחיש 1 (עם נתוני זרימה):

אחוז דיוק: 91.1% – המודל מזהה דפוסי תעבורה היטב, במיוחד עבור יישומים עם חתימות ברורות.

- יישומים בעלי תבנית ייחודית, כמו Zoom ו-YouTube, מזוהים בדיוק גבוה.
- Firefox ו-Google מציגים ביצועים נמוכים יותר בשל דפוסי תעבורה מגוונים.

תרחיש 2 (ללא נתוני זרימה):

אחוז דיוק: 87.6% – למרות אובדן נתוני זרימה, המודל עדיין שומר על דיוק גבוה יחסית.

- המודל מתקשה להבדיל בין היישומים, כפי שניתן לראות בירידה בערכי הדיוק והזיהוי.
- דיוק התחזיות עבור Zoom ו-YouTube נשאר גבוה יחסית בשל חתימות תעבורה ייחודיות.

<u>ניסוי בונוס: ניתוח זיהוי יישומים עם מקורות מעורבים</u>

מטרת הניסוי

בניסוי זה נבדק כיצד מודל מבוסס Random Forest, שאומן על נתוני תעבורה ממקור יחיד, מתפקד כאשר הוא מתבקש לזהות יישומים מתוך מקורות מעורבים.

החלוקה בוצעה כך ש-80% מהנתונים נלקחו ממקור אחיד (למשל רק YouTube), ואילו 20% מהנתונים לצורך הבדיקה הכילו שני מקורות (למשל YouTube ו-Spotify יחד).

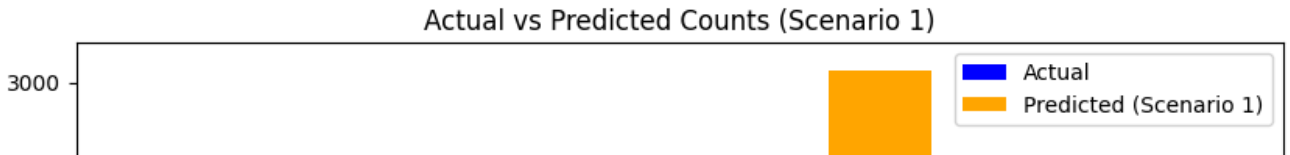
שיטות הערכה

המודל נבדק בשני התרחישים מהסעיף הקודם עם אותם מדדי הערכה.

תוצאות ופרשנות

תרחיש 1 (עם נתוני תוצאות ופרשנות)

אחוז דיוק: 71.5%

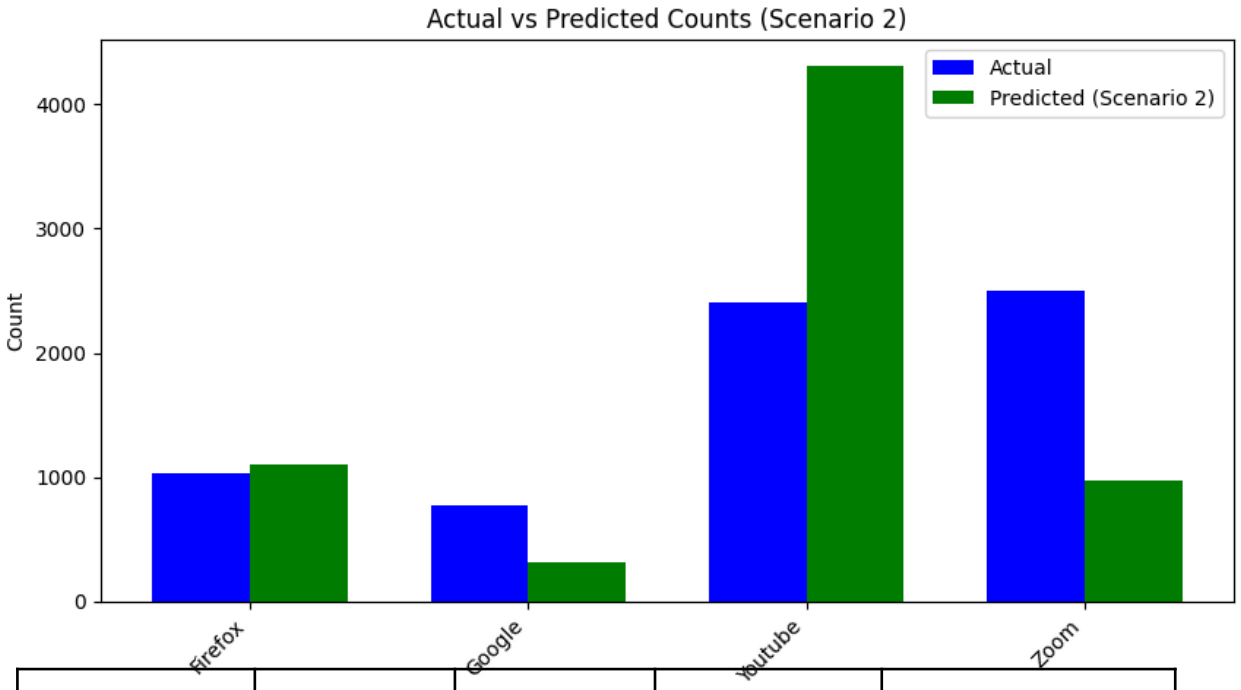


Application	Precision	Recall	F1-Score	Support
Firefox	75%	69%	72%	1036
Google	67%	42%	52%	770
Youtube	64%	82%	72%	2401
Zoom	81%	71%	76%	2501

Application	Actual	Predicted
Firefox	15%	14%
Google	11%	7%
Youtube	36%	46%
Zoom	37%	33%

תרחיש 2 (ללא נתוני זרימה)

אחוז דיוק: 53.5%



Application	Precision	Recall	F1-Score	Support
Firefox	62%	67%	64%	1036
Google	42%	17%	24%	770
Youtube	49%	88%	63%	2401
Zoom	67%	26%	38%	2501

Application	Actual	Predicted
Firefox	**15%**	**17%**
Google	**11%**	**5%**
Youtube	**36%**	**64%**
Zoom	**37%**	**15%**

ניתוח תוצאות

תרחיש 1 (עם נתוני זרימה):

דיוק גבוה יחסית (71.57%) – למרות השוני בין מערכי הנתונים, המודל עדיין מזהה דפוסי תעבורה טוב.

- זיהוי חזק של **YouTube ו-Zoom** , כנראה בגלל שלשניהם יש מאפייני תעבורה ברורים, גם כאשר Spotify מעורב.
- **Firefox ו-Google** מזוהים פחות טוב, ככל הנראה בגלל שלמודל יותר קשה לזהות דפוסים של תעבורה מעורבת.

תרחיש 2 (ללא נתוני זרימה):

דיוק ירד משמעותית (53.50%) – ללא נתוני זרימה, המודל מאבד מידע חשוב על מבנה הרשת, במיוחד כשתעבורה מעורבת.

- **YouTube** מזוהה בצורה טובה, הסיבה לכך היא שהמודל למד רק את דפוסי התעבורה של מקור יחיד, ולכן מנסה להתאים את כל הנתונים למקור זה.
- **Google ו-Zoom** מזוהים גרוע, מאחר שהמודל לא נתקל בתעבורה מעורבת באימון, הוא מתקשה בהתאמה.
- **Firefox מזוהה באופן דומה,** מה שמעיד על כך של-Firefox יש תעבורה יציבה יותר.

ביבליוגרפיה

סרטונים שנעזרנו בהם:

1. פיצוח הצפנה של פקטות בתעבורת רשת (עבודה עם מפתחות) ב- Wireshark:

How to DECRYPT HTTPS Traffic with Wireshark

2. למידת מכונה - אלגוריתם Random Forest:

What is Random Forest?

3. ניתוח תוצאות ונתוני הגרפים - Precision, Recall, & F1 Score Intuitively:

https://youtu.be/8d3JbbSj-I8?si=PacqCHK6i8RkIFm6

4. פעולות Regex בפייתון:

[5 Minute Tutorial] Regular Expressions (Regex) in Python

5. למידת מכונה - חלוקת העבודה לאימון המודל ובדיקתו:

Train Test Split with Python Machine Learning (Scikit-Learn)

6. עבודה עם ספריית Matplotlib בפייתון:

Matplotlib Tutorial (Part 2): Bar Charts and Analyzing Data from CSVs

ספריות ב - python שהשתמשנו בהן:

- **שאלה 3 חלק א':**
  - pandas
  - numpy
  - matplotlib.pyplot
  - re

pip install pandas numpy matplotlib

- **שאלה 3 חלק ב' (תוקף)**
  - pandas
  - numpy
  - matplotlib.pyplot
  - sklearn.model_selection >> train_test_split
  - sklearn.ensemble >> RandomForestClassifier
  - sklearn.metrics >> accuracy_score, classification_report
  - sklearn.preprocessing >> LabelEncoder
  - re

pip install pandas numpy matplotlib scikit-learn

- **שאלת בונוס**
  - pandas
  - numpy
  - matplotlib.pyplot
  - sklearn.ensemble >> RandomForestClassifier
  - sklearn.metrics >> accuracy_score, classification_report
  - re

pip install pandas numpy matplotlib scikit-learn

- **גרסת הפייתון שבה השתמשנו**

$ python3 - -version

Python 3.11.9

שאלות לבינה מלאכותית (ChatGPT):

- How can I use regex in python?
- Explain about Random Forest.
- How can I decrypt HTTP Wireshark packets?
- Is it better to use KNN than Random Forest?

- What is the KNN algorithm?

משתמשי לינקדאין:

- **Ofek Bar Shalom**
- **Amit Danino**
- **Roy Naor**
- **jesse briggs**