# רשתות תקשורת - פרויקט גמר

ת"ז : 213121296, 325763498, 325877124, 213525512

הקדמה :
**QUIC** (Quick UDP Internet Connections) הוא פרוטוקול העברת נתונים חדש יחסית, שתוכנן על ידי גוגל במטרה לשפר את חוויית הגלישה באינטרנט. המטרה המרכזית של QUIC היא להחליף את פרוטוקול TCP הוותיק, ולהפחית את זמן התגובה של חיבורי אינטרנט על ידי שילוב בין פרוטוקול העברה, אבטחת חיבור, ובקרת גודש.

## עקרונות עיקריים של:QUIC

1. **שימוש UDP ב-QUIC:** פועל על בסיס פרוטוקול UDP (User Datagram Protocol), המאפשר העברת נתונים ללא צורך בהגדרת חיבור כמו ב-TCP. דבר זה מאפשר התחלת העברת נתונים במהירות רבה יותר.
2. **שילוב בין אבטחה וחיבור :**פרוטוקול QUIC משלב את תהליך האבטחה (שכבה 7) עם תהליך הגדרת החיבור, כך שחיבור מאובטח נוצר כבר בשליחת החבילה הראשונה, ללא צורך בלחיצות יד נוספות.
3. **מזהי חיבור קבועים QUIC :**משתמש במזהי חיבור שאינם תלויים בכתובות ה-IP, מה שמאפשר לשמור על חיבור קיים גם אם כתובת ה-IP של אחד הצדדים משתנה.
4. **מניעת חסימת ראש התור (HOL):** בניגוד ל-TCP שבו אובדן חבילה בודדת יכול לחסום את כל הנתונים העוקבים, QUIC מאפשר העברת נתונים מתתי-זרמים שונים בצורה בלתי תלויה, כך שאובדן חבילה בתת-זרם אחד לא ישפיע על העברת הנתונים בתת-זרמים אחרים.
5. **ביצועים משופרים :**פרוטוקול QUIC משלב בקרת גודש מתקדמת עם אמינות העברת נתונים, מה שמביא לביצועים טובים יותר ברשתות בעלות רוחב פס נמוך או עם בעיות גודש.

בגדול QUIC תוכנן כדי להיות מהיר יותר, יעיל יותר ומאובטח יותר מפרוטוקולים קודמים, והוא משפר את חוויית המשתמש באינטרנט.

## חלק א – חלק "יבש" :

### שאלה 1 :

1. הקשר שבין CC (בקרת עומס) לשליחה אמינה : שניהם משתמשים בחלון בגודל מסוים שמגביל את כמות הפקטות שניתן לשלוח לפני קבלת ACK. כוונת החלון במקור הייתה לשמש את מנגנון ה-Flow-Control.
הבעיה היא שכאשר פקטה נאבדת, החלון לא מאפשר שידור נוסף של חבילות עד שהפקטה שנאבדה נשלחת שוב ומתקבל ACK.
מצב כזה יכול לגרום לחוסר יעילות, במיוחד במקרה שבו כל הפקטות שבחלון כבר נשלחו לרשת, כלומר הרשת לא תמשיך להיות בשימוש עד לקבלת ה-ACK למרות שיש לה קיבולת לנצל.
מספר הפקטות עליהן לא קיבלנו ACK לא בהכרח משקף את מספר הפקטות בתוך הרשת.

פותחה טכניקה בשם "window inflation and deflation" לפתרון הבעיה אך היא אינה אופטימלית.

2. בעיית Head of line blocking: מאחר ו-TCP מוודאת שליחה רציפה של נתונים (ע"פ סדר ה-Seq number), במקרה שפקטה נאבדת, היא תחסום את המעבר של כל הפקטות שאחריה עד שאותה פקטה תתקבל. ההבדל בין בעיה 1 ל-2 הוא שבבעיה אחת אנחנו מדברים על הצד השולח, שלא מגדיל את חלון השליחה, וב-2 אנחנו מדברים על הצד המקבל שלא יכול לקבל פקטות מעבר לזו שאבדה.

3. עיכובים עקב הגדרת חיבור: בכל פעם שנקודת קצה רוצה לתקשר דרך TCP נדרשת לחיצת ידיים משולשת לשם יצירת הקשר לפני שניתן לשלוח נתונים. ואם גם יש רצון להצפנת הקשר על ידי TLS יש צורך בעוד מעבר בין שתי נקודות הקצה על מנת לאבטח את התקשורת.

4. הגבלות עקב Header בגודל קבוע: ה-Header של TCP בגודל קבוע של 20 בתים שכולל בתוכו שדות לניהול התקנת חיבור, ניתוק, איפוס ואישור נתונים. שלושה שדות ספציפים ב-Header הופכים למגבלות ככל שמהירות הרשת עולה. שדה ה-sequance number, שדה הACK ושדה חלון ה-Flow control. במהירויות גבוהות ניתן למצות את ה- sequance numbers מאחר והשדה שלהם בגודל 4 בתים ניתן להגיע רק עד מספר מסויים ואז יש צורך בחזרה על אותם מספרים - מה שיכול ליצור בלבול. כמו כן מספר חלון flow-control קטן יכול להאט את השליחה.
עצם זה שה-Header קבוע עלול להגביל את העברת הנתונים ברשת.

5. מזהה חיבור ייחודי וכתובת IP: כתובת ה-IP של כל אחת מנקודות הקצה יכולה להשתנות בחיבור TCP משלל סיבות. מאחר ו-TCP משתמש בחיבור של ה-IP ומספר הפורט של שתי נקודות הקצה בתור מזהה החיבור, כל שינוי ב-IP יכול לשבור את החיבור הקיים והנתונים עלולים להיאבד. כדי לחזור לקשר יש צורך בלחיצת יד משולשת בשנית כדי ליצור קשר חדש.

שאלה 2:

1. מסירה אמינה: על הפרוטוקול לוודא כי הנתונים שנשלחו ליעד מגיעים במלואם ובסדר הנכון. כולל מנגנונים למניעת שגיאות, שידור חוזר של פקטות שאבדו ועוד. ניתן לראות ש-SCTP ו-TCP משתמשים ב-sequance numbers לצורך זה.

2. מנגנון "Congestion Control" (בקרת עומס): על הפרוטוקול להיות מסוגל לנהל את זרימת הנתונים ברשת בצורה טובה כך שימנע עומס על הרשת. הוא עושה זאת ע"י התאמת קצב שליחת הנתונים לקיבולת של הרשת. ניתן לראות ש- SCTP ו-TCP משתמשים במנגנון זה.

3. תמיכה בתקשורת בזמן אמת: על הפרוטוקול להיות מסוגל להעביר נתונים בזמן אמת כמו שיחת וידאו וכדומה. כאשר הפרוטוקול מתעדף מהירות ע"ג אמינות. פרוטוקול RTP נועד לצורך זה.

4. טיפול יעיל בחיבורים: שימור החיבור על מנת להימנע מהעיכוב שגוררת לחיצת ידיים משולשת. פרוטוקול T/TCP מממש זאת.

5. גמישות: על הפרוטוקול לתמוך בסביבות תקשורת שונות ודרישות רשת מגוונות. לדוגמא SCTP מאפשרת שינוי של IP תוך כדי הקשר ותומך בריבוי זרימות בקשר יחיד.

שאלה 3:

בחירת ID לכל אחד מהצדדים-שלב ראשון הקמת חיבור מאובטח באמצעותRTT אחד. החבילה הראשונה (Initial packet) משמשת לקבוע איך כל צד יזוהה במהלך החיבור. כל צד בוחר ומציין בחבילה הראשונית שלו את ID שהוא מעדיף להשתמש בו. הצד השני, מקבל אותו וזה יהיה כתובת

היעד שלו לחבילות נוספות. אימות כתובת לקוח- לאחר מכן, אם ירצה הוא יכול לאמת אמת את כתובת ה-IP של הלקוח באמצעות Retry packet. כך הלקוח צריך לשלוח שוב את החבילה הראשונית שלו, הפעם עם הסיסמה שקיבל מהשרת, כדי להמשיך בתהליך כינון החיבור. וזה מוסיף לאבטחה ומונע התחזיות. אבטחת TLS 1.3 -בתוך החבילה הראשונית, QUIC כולל גם הודעות עבור תהליך האבטחה של TLS 1.3. שמאפשר להצפין תוכן חבילות שישלחו כל שלא יחשפו ולא ישונו. מה ששומר על החבילות לאחר ההעברה שלמות ומקוריות. אימות פרמטרי תעבורה- במהלך תהליך האבטחה של TLS, גם פרמטרי התעבורה עוברים אימות כדי לוודא ששני הצדדים מסכימים על ההגדרות. מה שמגביר יעילות העברה ומונע טעויות בתקשורת וקשיי תאימות בין שני הצדדים.

פתרון בעיה 3- נשים לב שQUIC מאפשר ללקוח לשלוח O-RTT נתוני יישום מוצפנים כבר בחבילה הראשונה אבל זה לא בטוח כל כך.
ב TCP יש שתי לחיצות ידיים נפרדות: הראשונה דואגת לאבטחה באמצעות TLS והשניה קובעת את הפרמטרים עבור הנתונים עצמם( גודל חבילה למשל) כל לחיצת יד דורשת תקשורת הלוך וחזור (RTT) אחת, כך שלוקח לפחות שני סבבים (RTTs) להקים חיבור מאובטח לחלוטין. ואילו בQUIC משלבים את שתי לחיצות היד לתהליך אחד בתוך הודעה ראשונית אחת בRTT. כך ש שגם החיבור מאובטח וגם ניתן המידע הדרוש להעברת נתונים וזה מקצר משמעותית את הזמן הנדרש להקמת חיבור מאובטח.
בנוסף, בניגוד ל-TCP, שעובד על שילוב של כתובות IP ומספרי פורט של שני הקצוות החיבור כמזהה חיבור, חיבורי QUIC יכולים לשרוד שינויים בכתובות הבסיסיות של הפרוטוקול בזכות שימוש ב-Connection ID (מזהה חיבור ולכן אם רוצים למשל לעבור מרשת WI-FI לרשת סלולרית זה אפשרי .לאחר שינוי רשת, קצה נודד יכול לשלוח חבילה עם מזהי חיבור שנקבעו מראש באמצעות הכתובת החדשה שלו כדי לאתחל את תהליך נדידת החיבור. לאחר שהקצה השני מקבל את החבילה הזאת, הוא יבצע אימות נתיב כדי לאמת את הבעלות של העמית על הכתובת החדשה על ידי שליחת מסגרת אתגר מיוחדת המכילה נתונים אקראיים לכתובת החדשה של העמית והמתנה לתגובה מהדהדת עם אותם נתונים, ושני הקצוות יכולים להמשיך להחליף נתונים לאחר אימות הכתובת החדשה.

שאלה 4:

מבנה החבילה של QUIC: (פתרון בעיה מס 4-)במקום כמו TCP שבו הכותרת קבועה, ל-QUIC יש שני סוגים של כותרות חבילה: long header, short header. כדי להקים חיבור נשתמש בפורמט כותרת ארוך שיכיל פרטים של מידע שנחוצים ללחיצת הידיים הראשונית. לאחר החיבור, רק שדות כותרת מסוימים נחוצים ולכן נשתמש בפורמט כותרת קצר בשביל הגברת היעילות. בפרוטוקול QUIC, בתוך כל חבילה יש מסגרות= מידע. מסגרות יכולות להכיל סוגים שונים של מידע כמו נתוני משתמש, בקשות, תגובות, או מידע על בקרת זרימה. כל חבילה יכולה להכיל מסגרת אחת או יותר, וזה מאפשר ל-QUIC להיות מאוד גמיש ויעיל, כיוון שהוא יכול לשלוח סוגים שונים של מידע במנה אחת במקום להפריד אותם למנות שונות.
לכל חבילה בחיבור QUIC מוקצה מספר חבילה ייחודי. מספר זה גדל באופן מונוטוני ומציין את סדר השידור של החבילות בלי קשר למקרה שחבילה "נאבדת". לכן ניתן להשתמש בו כדי לדעת בקלות ובמדויק כמה חבילות עשויות להיות ברשת, בהשוואה לבקרת העומס של TCP שמשתמשת באותו חלון בקרת זרימה עבור אמינות.
המקבל ב QUIC מאשר ACKs את מספר החבילה הגדול ביותר שהתקבל עד כה, יחד עם ACK סלקטיבי (מאשר את כל החבילות שהתקבלו מתחתיו, מקודד בטווחי מספרי חבילות רציפים).
השימוש במסגרות ACK מוגדרות בכוונה יכול לתמוך עד 256 מחסומי ACK במסגרת ACK אחת, בהשוואה ל3- טווחי SACK של TCP עקב מגבלה בגודל שדה האפשריות של TCP. זה מאפשר ל-QUIC לאשר חבילות שהתקבלו מספר פעמים במסגרות ACK מרובות, מה שמוביל לחוסן גבוה יותר נגד סידור מחדש ואובדן של חבילות. כאשר חבילת QUIC מקבלת ACK, זה מציין שכל המסגרות שנישאו בחבילה התקבלו.

פתרון בעיה 1- כדי למנוע צמצום מיותר של חלון העומס כמו שקורה ב TCP. QUIC ,לא מקטינה את חלון העומס אלא אם היא מזהה עומס יתר עקבי. כאשר שני חבילות הדורשות אישור מוכרזות כאבודות, עומס יתר עקבי ייקבע אם אף אחת מהחבילות שנשלחו ביניהן לא מאושרת, דוגמת RTT הייתה קיימת לפני שנשלחו וההפרש בין זמן השליחה שלהן עולה על משך העומס העקבי המחושב על סמך זמן הלוך ושוב הממוצע (rtt_smoothed), הסטייה של דוגמאות ה-RTT (rttvar) והזמן המקסימלי שהמקבל עשוי לעכב את שליחת האישור .שולח QUIC יאט את השליחה שלו כדי להפחית את הסיכויים לגרום לעומס לטווח קצר על ידי הבטחת מרווח השליחה בין החבילות יעלה על מגבלה המחושבת על סמך זמן הלוך ושוב הממוצע (rtt_smoothed),גודל חלון העומס וגודל החבילה.

פתרון בעיה 2- ב-QUIC יכולים להתקיים מספר זרמים על חיבור TCP אחד, לכל חיבור QUIC יכולות להיות זרימות בו-זמניות מרובות .בנוסף QUIC ,אימץ גם את הרעיון של חיתוך נתונים למסגרות ומשתמש בהן כיחידת התקשורת הבסיסית. מכיוון ש-QUIC משתמש בזרמים עצמאיים מרובים, הוא נמנע מבעיית חסימת ראש התור הנגרמת מהמתנה לשחזור חבילות שאבדו ב-TCP. כאשר חבילה אובדת, רק הזרמים עם מסגרות נתונים הכלולות בחבילה יצטרכו להמתין לשידור חוזר של המסגרות האבודות. זה לא יחסום זרמים אחרים מלהתקדם כי כל זרם נושא נתונים עצמאיים ויכול להיות מטופל באופן בלתי תלוי.

פתרון בעיה 5-לאחר שינוי כתובת IP בשימוש ב-TCP : החיבור הישן ייפסל ויידרש לחיצת יד משולשת חדשה שתארך -1) Time Trip-Round) RTT לפני שניתן יהיה להחליף נתוני יישום .לעומת זאת, בשימוש ב-QUIC : :נוכל לעשות שימוש חוזר בחיבור הישן והצדדים יכולים להתחיל לשלוח נתונים ישירות אם השרת אימת את כתובת הלקוח בעבר (לדוגמה, כאשר הלקוח חוזר לכתובת ישנה).

שאלה 5 :

QUIC משתמש בזיהוי אובדן מבוסס ACK כדי להבחין בין packets מושהות לpackets אבודות.

אם packet נשארת ללא אישור בזמן שהpackets הבאות מאושרות, וערכי סף מסוימים מתקיימים, QUIC מכריז על הפריימים בחבילה הלא מאושרת כאבדה. ערכי סף אלה מבוססים על numbers packets וזמן, המאפשרים חלון זמן מסוים לפני שpackets לא מאושרות ייחשבו כאבדות וישודרו מחדש, ובכך מונעות שידורים חוזרים מיותרים.

כדי לזהות אובדן של tail packets,

QUIC מאתחל טיימר בדיקה (PTO) המבוסס על וריאציה משוערת של וריאציית RTT network ו-RTT varition. כאשר הטיימר יפוג, QUIC שולח חבילת בדיקה כדי לשדר מחדש כמה נתונים ולהפחית את הצורך בשידורים חוזרים מיותרים.

לאחר זיהוי אובדן, QUIC מכניס frames שאבדו לpackets יוצאות חדשות עם מספרי packets חדשים. מנגנון זיהוי ושחזור אובדן זה מאפשר ל-QUIC לתמוך באספקה אמינה של זרם בתים מסודר בדומה ל-TCP.

שאלה 6 :

בקרת עומס ב-QUIC הוא אספקט מאוד נחוץ מכיוון שהוא משפיע באופן ישיר על אפקטיביות ואמינות המידע. כאשר אנו נשתמש ב-QUIC, איכות המידע עלולה להיפגע מאובדן החבילות, זמן השהייה ובעיות בזמינות רוחב פס. ביצוע בקרת עומס יעזור לנו להימנע מעומס יתר על השרת ויבטיח לנו שזרם המידע יתאים את עצמו למצב הנוכחי של הרשת.

ישנם כמה עקרונות מפתח לבקרת עומס ב-QUIC:
עקרונות בסיסיים של UDP עם השפעה של TCP - האופן בו QUIC בנוי על UDP אך משתמש בחידושים ואסטרטגיות שנלקחו מTCP. לדוגמה מכניקות כמו "התחלה איטית", הימנעות מעומס, שידור חוזר מהיר והתאוששות מהירה.

איתור ושיקום אובדן מידע - QUIC בנוי בצורה שמכילה מכניקות עבור זיהוי אובדן חבילות, בדרך כלל על ידי זיהוי חריגה בפסקי הזמן של שידורים חוזרים (RTOs) או על ידי ביצוע אישורים כפולים, בדומה לTCP. האמנם, QUIC משפר עקרונות אלו באמצעות אישורים וpacket number spaces מדויקים יותר המאפשרים זיהוי אובדן מהיר ומדויק יותר.

העברת path - מכיוון ש QUIC מנהל את כל החיבורים כאחד ולא כסטרימים בודדים, הוא יכול להתאים את קצב השליחה שלו בהתבסס על תנאים כלליים ולא בהכרח להתחשב בכל סטרים בנפרד. ניהול זה עוזר למנוע השתלטות של זרם אחד על משאבי החיבור.

ריבוי ללא Head of Line Blocking - השימוש של QUIC במספר סטרימים דרך אותו חיבור מאפשר לו להתעלם מבעיית הHead of Line Blocking הקיימת בTCP. המשמעות היא שאובדן חבילות בזרם אחד לא ישפיע כלל על העברת הנתונים בסטרים אחר, מה שמאפשר QUIC שמור על יעילות ותגובתיות כללית גבוהה יותר.

יישום של QUIC יתחיל לעיתים קרובות באלגוריתמים לבקרת עומס כגון:

Cubic - ספק ברירת מחדל לבקרת עומס עבור יישומי TCP ו-QUIC רבים. CUBIC תוכנן להיות אגרסיבי בסביבות עם מוצרי עיכוב רוחב פס גדולים ותוך כדי נשאר הוגן לפרוטוקולים אחרים כמו TCP.

Reno - אלגוריתם בקרת עומס בTCP שתוכנן בעבר על ידי הגדלת חלון העומס באופן ליניארי, שעשוי להתאים פחות לרשתות עם רוחב פס גבוה או זמן שהייה גבוה, אך הוא פשוט יותר משיטות אחרות ומובן בקלות.

BBR (Bottleneck Bandwidth and Round-trip propagation time) - אלגוריתם הBBR של גוגל, המשמש בכמה יישומי QUIC, ממדל את השרת על ידי אומדן רציף של round-trip time המינימלי של צוואר הבקבוק, ומאפשר למקסם את תפוקת הרשת תוך מזעור זמן השהייה.

בעוד שעקרונות אלו מספקים לנו בסיס תיאורטי לבקרת עומס ב-QUIC, חשוב לזכור שדברים אלו משתנים לפני היישום עצמו. לדוגמה, ספריות ספציפיות כמו aioquic יכולות לשלוט על הבקרה ולספק אותה על ידי הספרייה עצמה. כלומר, המשתמש אינו צריך ליישם דבר אלא להגדיר.

## חלק ב – חלק "רטוב":

בחרנו להתמקד בעבודה זו בנושא ריבוי זרימות. בקוד שכתבנו יש מספר מחלקות שונות כך שלכל אחת יהיה תפקיד ייחודי משלה להפעלת כל תהליך הזרימה:

### 1. מחלקת Packet

**מטרת המחלקה:**

- לייצג חבילה של נתונים שמועברת בין הלקוח לשרת.

**שדות:**

- stream_id : מזהה את ה stream-שאליו שייכת החבילה.
- seq_no :מספר הסדר של החבילה ב.stream-
- data :הנתונים עצמם (בייטים) שמועברים בתוך החבילה.

**פונקציות:**

- __init__(self, stream_id, seq_no, data) : קונסטרקטור ליצירת אובייקט חבילה עם מזהה stream, מספר סדר, ונתונים.
- encode(self) : מקודדת את החבילה למחרוזת שניתן לשלוח אותה ברשת. הנתונים מקודדים במבנה טקסטואלי עם הפרדה באמצעות פסיקים.
- decode(data) : פונקציה סטטית שמפענחת מחרוזת שהתקבלה לאובייקט חבילה. מקבלת את המידע, מפרקת אותו למרכיבים ומחזירה אובייקט חבילה חדש.

### 2. מחלקת Connection

**מטרת המחלקה:**

- לנהל את החיבור לשרת ולשלוח את החבילות לכתובת השרת.

**שדות:**

- server_address : כתובת השרת שאליו יישלחו החבילות.
- socket : אובייקט של socket שמבצע את החיבור לשרת.

**פונקציות:**

- __init__(self, server_address=("127.0.0.1", 9999)) : קונסטרקטור ליצירת אובייקט Connection עם כתובת השרת ויצירת UDP Socket.
- send_packet(self, packet) :שולחת חבילה לשרת על ידי שימוש בפונקציית ה-send_to של ה-Socket.
- close(self): סוגרת את ה socket-כאשר הסיום של העבודה מתבצע.

### 3. מחלקת Stream

**מטרת המחלקה:**

- לקרוא נתונים מקובץ, לחלק את הנתונים לחבילות ולשלוח אותן.

**שדות:**

- stream_id : את ה.stream-
- file_path :נתיב הקובץ שממנו נקראים הנתונים.
- packet_size :גודל החבילות שנשלחות (נקבע באקראיות בטווח של 1000-2000 בתים).

- seq_no: מספר הסדר של החבילות.

**פונקציות:**

- __init__(self, stream_id, file_path): קונסטרקטור ליצירת אובייקט Stream עם מזהה stream, נתיב הקובץ וגודל החבילות.
- read_packet(self): קורא את הנתונים מהקובץ בקטעים בגודל ה-packet_size, יוצר חבילות עם הנתונים הללו ומשדר אותן.

**4. קובץ הלקוח (Client)**

**מטרת הקובץ:**

- לשלוח נתונים לשרת באמצעות מספר streams במקביל.

**פונקציות עיקריות:**

- generate_random_bytes(size): מייצרת נתונים אקראיים בגודל מסוים לצורך יצירת קבצים להעברה לשרת.
- send_stream(stream_id, file_path): יוצרת אובייקט של Stream, שולחת את הנתונים לשרת ומחשבת את הסטטיסטיקות (בתים וחבילות שנשלחו).
- main(): מקבלת מספר streams מהמשתמש, יוצרת קבצים עם נתונים אקראיים, ושולחת את הנתונים ב-threads נפרדים עבור כל Stream. לאחר מכן, מדפיסה סטטיסטיקות על הביצועים.

**5. קובץ השרת (Server)**

**מטרת הקובץ:**

- לקבל חבילות מהלקוח ולעבד אותן.

**פונקציות עיקריות:**

- handle_client(server_socket, stop_event): מאזינה לחבילות שמתקבלות מהלקוח, מדפיסה את המידע מהחבילות ומסיימת את העבודה כאשר מתקבלת חבילת סיום (END).
- main(): מגדירה את ה-Socket של השרת, מאזינה לחבילות, וקוראת לפונקציה handle_client להפעיל threads נפרדים לטיפול בקבצי הלקוח.

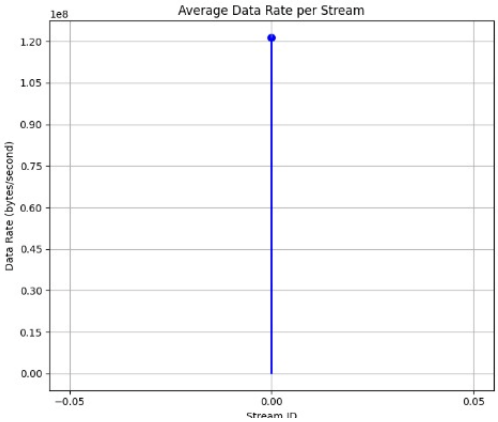
## ביצוע ניסויים והצגת גרפים :

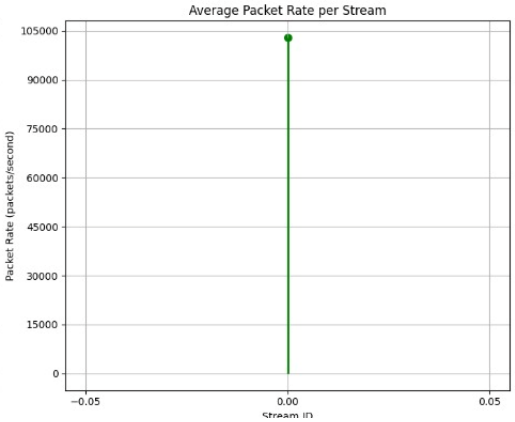
לאחר שכתבנו את הקוד, הרצנו ניסויים על מספר זרימות שמשתנה בין 1 ל-10.

מצורפים צילומי מסך של הדפסות הסטטיסטיקות עבור כל זרימה בנפרד ועבור כל הזרימות יחד.

הרצנו זרימה אחת :

1. מספר הבתים שעברו בסה"כ בכל הזרימות : 2097152
2. מספר החבילות שעברו בסה"כ בכל הזרימות :1778
3. קצב הנתונים הכולל, כלומר : כמה בתים הגיעו ליעד בשנייה, בממוצע : 121330441.57
4. קצב החבילות הכולל, כלומר : כמה חבילות הגיעו ליעד בשנייה, בממוצע : 102865.95

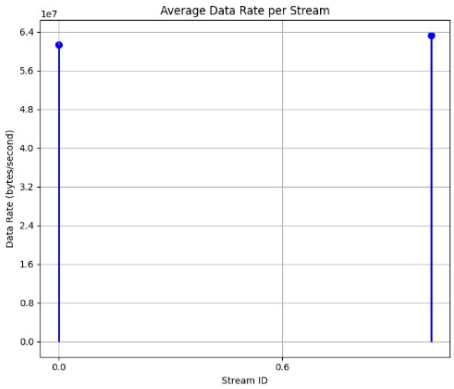


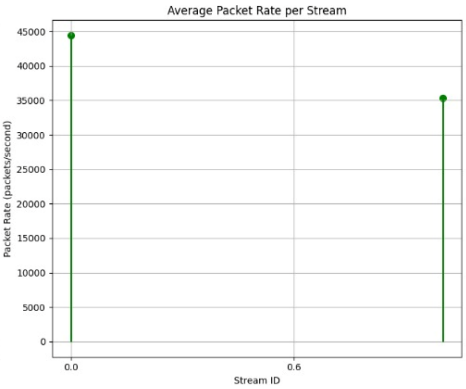




```
Please enter the number of streams you want: 1
Data for stream 0 saved to stream_data_0.bin
Stream 0: Completed sending stream_data_0.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1778
  Data rate: 121330441.57 bytes/second
  Packet rate: 102865.95 packets/second

Overall Statistics:
Total bytes sent: 2097152
Total packets sent: 1778
Average data rate: 121330441.57 bytes/second
Average packet rate: 102865.95 packets/second

Process finished with exit code 0
```

הרצנו 2 זרימות :

1. מספר הבתים שעברו בסה"כ בכל הזרימות : 4194304
2. מספר החבילות שעברו בסה"כ בכל הזרימות :2689
3. קצב הנתונים הכולל, כלומר : כמה בתים הגיעו ליעד בשנייה, בממוצע : 62278736.33
4. קצב החבילות הכולל, כלומר : כמה חבילות הגיעו ליעד בשנייה, בממוצע : 39927.37

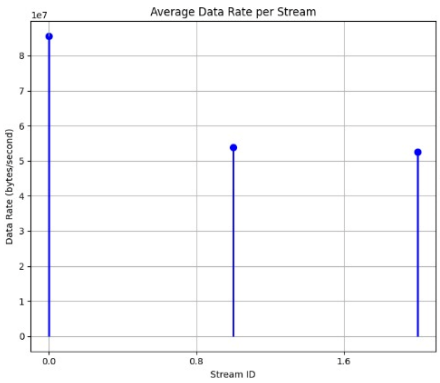


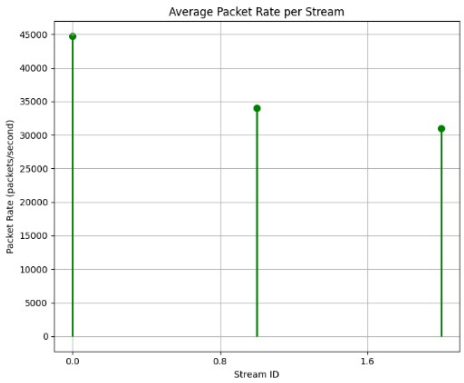




```
Please enter the number of streams you want: 2
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Stream 1: Completed sending stream_data_1.bin
Stream 0: Completed sending stream_data_0.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1519
  Data rate: 61323738.10 bytes/second
  Packet rate: 44417.74 packets/second
Stream 1:
  Bytes sent: 2097152
  Packets sent: 1170
  Data rate: 63263949.58 bytes/second
  Packet rate: 35294.92 packets/second

Overall Statistics:
Total bytes sent: 4194304
Total packets sent: 2689
Average data rate: 62278736.33 bytes/second
Average packet rate: 39927.37 packets/second
```

הרצנו 3 זרימות:

1. מספר הבתים שעברו בסה"כ בכל הזרימות: 6291456
2. מספר החבילות שעברו בסה"כ בכל הזרימות: 3655
3. קצב הנתונים הכולל, כלומר: כמה בתים הגיעו ליעד בשנייה, בממוצע: 60912381.25
4. קצב החבילות הכולל, כלומר: כמה חבילות הגיעו ליעד בשנייה, בממוצע: 35386.84

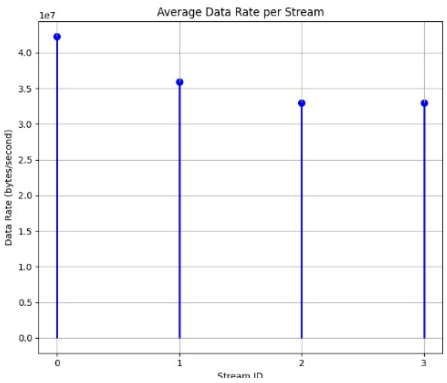


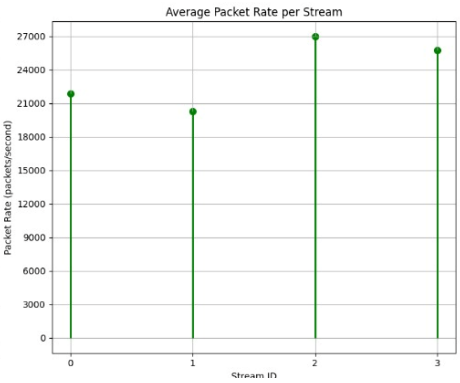




```
Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1096
  Data rate: 85540144.14 bytes/second
  Packet rate: 44704.44 packets/second
Stream 1:
  Bytes sent: 2097152
  Packets sent: 1323
  Data rate: 53903574.06 bytes/second
  Packet rate: 34005.37 packets/second
Stream 2:
  Bytes sent: 2097152
  Packets sent: 1236
  Data rate: 52606638.69 bytes/second
  Packet rate: 31004.81 packets/second

Overall Statistics:
Total bytes sent: 6291456
Total packets sent: 3655
Average data rate: 60912381.25 bytes/second
Average packet rate: 35386.84 packets/second
```

הרצנו 4 זרימות:

1. מספר הבתים שעברו בסה"כ בכל הזרימות: 8388608
2. מספר החבילות שעברו בסה"כ בכל הזרימות: 5628
3. קצב הנתונים הכולל, כלומר: כמה בתים הגיעו ליעד בשנייה, בממוצע: 35638116.59
4. קצב החבילות הכולל, כלומר: כמה חבילות הגיעו ליעד בשנייה, בממוצע: 23909.96

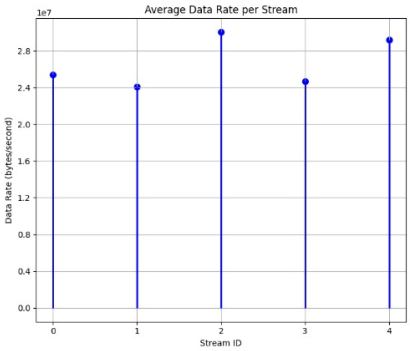


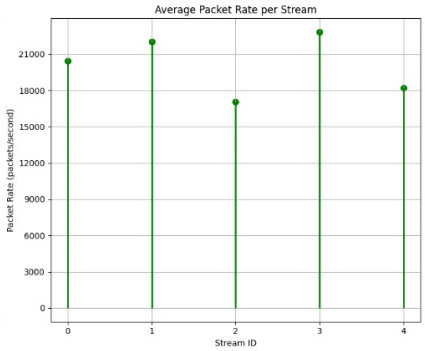




```
Please enter the number of streams you want: 4
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Stream 0: Completed sending stream_data_0.bin
Stream 1: Completed sending stream_data_1.bin
Stream 3: Completed sending stream_data_3.bin
Stream 2: Completed sending stream_data_2.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1086
  Data rate: 42263916.16 bytes/second
  Packet rate: 21886.16 packets/second
Stream 1:
  Bytes sent: 2097152
  Packets sent: 1185
  Data rate: 35894655.53 bytes/second
  Packet rate: 20282.35 packets/second
Stream 2:
  Bytes sent: 2097152
  Packets sent: 1717
  Data rate: 32938493.83 bytes/second
  Packet rate: 26967.71 packets/second
Stream 3:
  Bytes sent: 2097152
  Packets sent: 1640
  Data rate: 32938493.83 bytes/second
  Packet rate: 25758.33 packets/second

Overall Statistics:
Total bytes sent: 8388608
Total packets sent: 5628
Average data rate: 35638116.59 bytes/second
Average packet rate: 23909.96 packets/second
```

הרצנו 5 זרימות :

1. מספר הבתים שעברו בסה"כ ב כל זרימה : 10485760
2. מספר החבילות שעברו בסה"כ בכל זרימה :6049
3. קצב הנתונים הכולל, כלומר : כמה בתים הגיעו ליעד בשנייה, בממוצע : 2644771.76
4. קצב החבילות הכולל, כלומר : כמה חבילות הגיעו ליעד בשנייה, בממוצע : 20301.59




```
Please enter the number of streams you want: 5
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Stream 2: Completed sending stream_data_2.bin
Stream 4: Completed sending stream_data_4.bin
Stream 0: Completed sending stream_data_0.bin
Stream 3: Completed sending stream_data_3.bin
Stream 1: Completed sending stream_data_1.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1688
  Data rate: 25396104.07 bytes/second
  Packet rate: 20441.35 packets/second
Stream 1:
  Bytes sent: 2097152
  Packets sent: 1919
  Data rate: 24086083.07 bytes/second
```

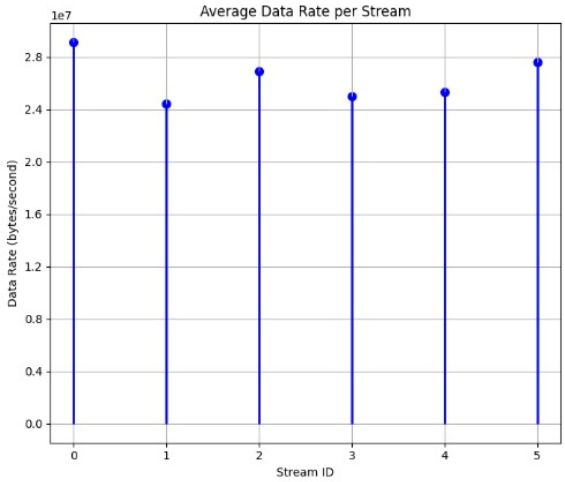
```
  Data rate: 24086083.07 bytes/second
  Packet rate: 22039.98 packets/second
Stream 2:
  Bytes sent: 2097152
  Packets sent: 1193
  Data rate: 30014444.12 bytes/second
  Packet rate: 17074.22 packets/second
Stream 3:
  Bytes sent: 2097152
  Packets sent: 1941
  Data rate: 24653694.43 bytes/second
  Packet rate: 22818.00 packets/second
Stream 4:
  Bytes sent: 2097152
  Packets sent: 1308
  Data rate: 29172115.73 bytes/second
  Packet rate: 18194.74 packets/second

Overall Statistics:
Total bytes sent: 10485760
Total packets sent: 8049
Average data rate: 26447701.76 bytes/second
Average packet rate: 20301.59 packets/second
```

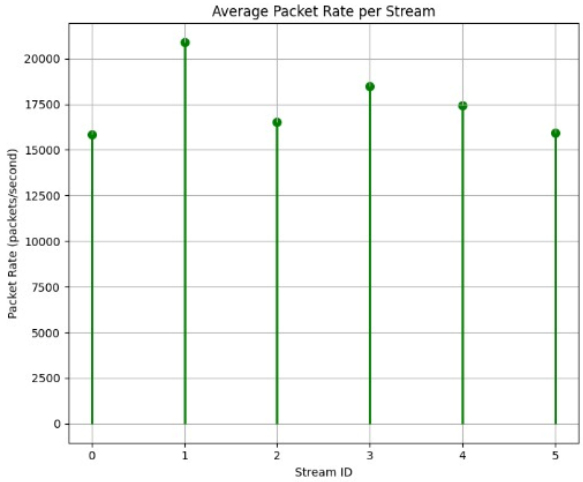
הרצנו 6 זרימות :

1. מספר הבתים שעברו בסה"כ בכל הזרימות : 12582912
2. מספר החבילות שעברו בסה"כ בכל הזרימות :8433
3. קצב הנתונים הכולל, כלומר : כמה בתים הגיעו ליעד בשנייה, בממוצע : 26271737.64
4. קצב החבילות הכולל, כלומר : כמה חבילות הגיעו ליעד בשנייה, בממוצע : 17607.18

```
Please enter the number of streams you want: 6
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Data for stream 5 saved to stream_data_5.bin
Stream 0: Completed sending stream_data_0.bin
Stream 5: Completed sending stream_data_5.bin
Stream 2: Completed sending stream_data_2.bin
Stream 4: Completed sending stream_data_4.bin
Stream 3: Completed sending stream_data_3.bin
Stream 1: Completed sending stream_data_1.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1141
  Data rate: 29095306.37 bytes/second
  Packet rate: 15829.92 packets/second
Stream 1:
  Bytes sent: 2097152
```

```
  Data rate: 26861005.91 bytes/second
  Packet rate: 16535.55 packets/second
Stream 3:
  Bytes sent: 2097152
  Packets sent: 1551
  Data rate: 24993018.80 bytes/second
  Packet rate: 18484.20 packets/second
Stream 4:
  Bytes sent: 2097152
  Packets sent: 1445
  Data rate: 25294737.00 bytes/second
  Packet rate: 17428.82 packets/second
Stream 5:
  Bytes sent: 2097152
  Packets sent: 1211
  Data rate: 27564627.09 bytes/second
  Packet rate: 15917.19 packets/second

Overall Statistics:
Total bytes sent: 12582912
Total packets sent: 8433
Average data rate: 26271737.64 bytes/second
Average packet rate: 17607.18 packets/second
```



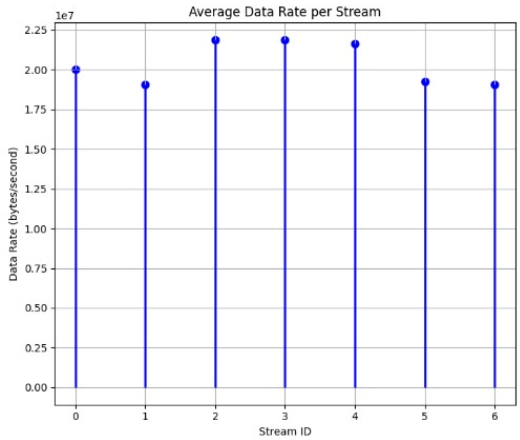



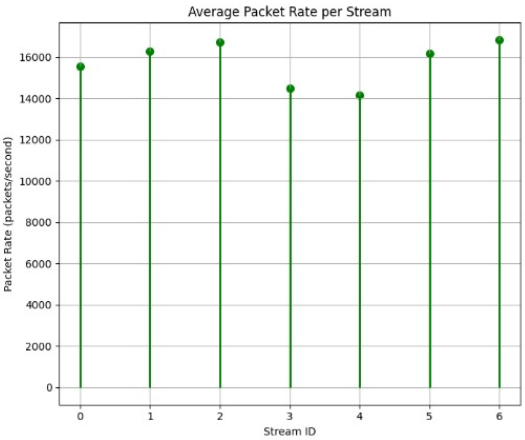

הרצנו 7 זרימות :

1. מספר הבתים שעברו בסה"כ בכל הזרימות : 14680064
2. מספר החבילות שעברו בסה"כ בכל הזרימות :11399
3. קצב הנתונים הכולל, כלומר : כמה בתים הגיעו ליעד בשנייה, בממוצע :20320584.47
4. קצב החבילות הכולל, כלומר : כמה חבילות הגיעו ליעד בשנייה, בממוצע : 15778.84

```
Please enter the number of streams you want: 7
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Data for stream 5 saved to stream_data_5.bin
Data for stream 6 saved to stream_data_6.bin
Stream 2: Completed sending stream_data_2.bin
Stream 3: Completed sending stream_data_3.bin
Stream 4: Completed sending stream_data_4.bin
Stream 0: Completed sending stream_data_0.bin
Stream 5: Completed sending stream_data_5.bin
Stream 1: Completed sending stream_data_1.bin
Stream 6: Completed sending stream_data_6.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1630
  Data rate: 19996574.12 bytes/second
  Packet rate: 15542.23 packets/second
```

```
  Packet rate: 14482.93 packets/second
Stream 4:
  Bytes sent: 2097152
  Packets sent: 1372
  Data rate: 21641586.60 bytes/second
  Packet rate: 14158.37 packets/second
Stream 5:
  Bytes sent: 2097152
  Packets sent: 1764
  Data rate: 19249869.29 bytes/second
  Packet rate: 16191.85 packets/second
Stream 6:
  Bytes sent: 2097152
  Packets sent: 1850
  Data rate: 19074586.62 bytes/second
  Packet rate: 16826.62 packets/second

Overall Statistics:
Total bytes sent: 14680064
Total packets sent: 11399
Average data rate: 20320584.47 bytes/second
Average packet rate: 15778.84 packets/second
```



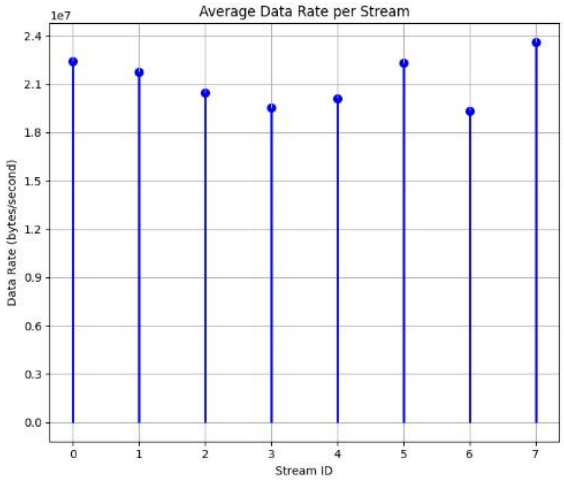



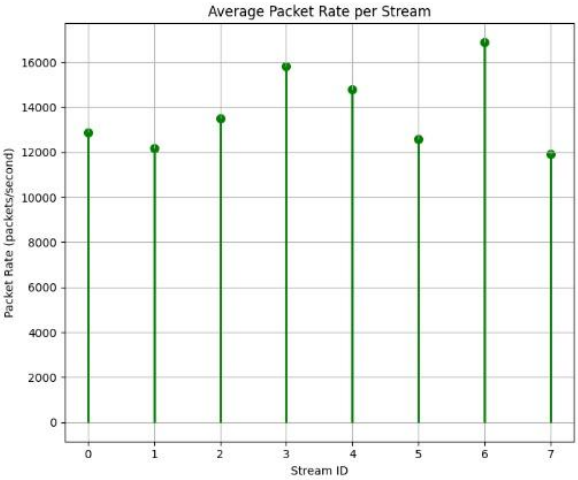
הרצנו 8 זרימות:

1. מספר הבתים שעברו בסה"כ בכל הזרימות: 16777216
2. מספר החבילות שעברו בסה"כ בכל הזרימות: 11076
3. קצב הנתונים הכולל, כלומר: כמה בתים הגיעו ליעד בשנייה, בממוצע: 21090334.35
4. קצב החבילות הכולל, כלומר: כמה חבילות הגיעו ליעד בשנייה, בממוצע: 13923.44

```
Please enter the number of streams you want: 8
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Data for stream 5 saved to stream_data_5.bin
Data for stream 6 saved to stream_data_6.bin
Data for stream 7 saved to stream_data_7.bin
Stream 0: Completed sending stream_data_0.bin
Stream 7: Completed sending stream_data_7.bin
Stream 5: Completed sending stream_data_5.bin
Stream 1: Completed sending stream_data_1.bin
Stream 2: Completed sending stream_data_2.bin
Stream 4: Completed sending stream_data_4.bin
Stream 3: Completed sending stream_data_3.bin
Stream 6: Completed sending stream_data_6.bin

Stream Statistics:
Stream 0:
  Bytes sent: 2097152
  Packets sent: 1204
```

```
  Packet rate: 14802.46 packets/second
Stream 5:
  Bytes sent: 2097152
  Packets sent: 1181
  Data rate: 22320350.54 bytes/second
  Packet rate: 12569.59 packets/second
Stream 6:
  Bytes sent: 2097152
  Packets sent: 1830
  Data rate: 19341041.72 bytes/second
  Packet rate: 16877.23 packets/second
Stream 7:
  Bytes sent: 2097152
  Packets sent: 1059
  Data rate: 23587899.99 bytes/second
  Packet rate: 11911.19 packets/second

Overall Statistics:
Total bytes sent: 16777216
Total packets sent: 11076
Average data rate: 21090334.35 bytes/second
Average packet rate: 13923.44 packets/second
```



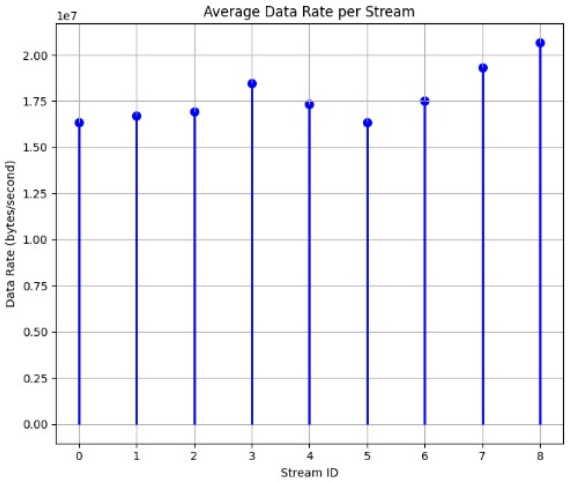



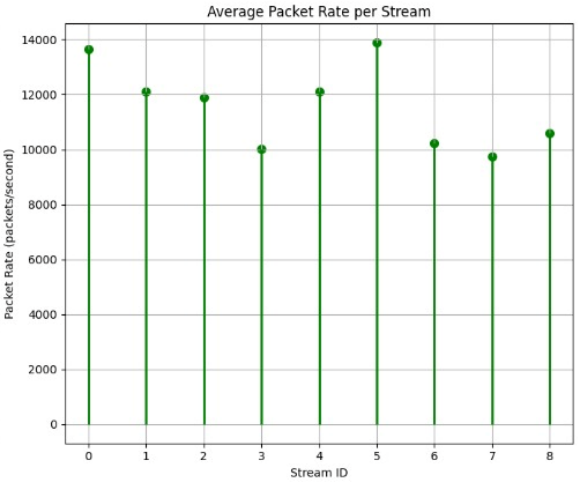
הרצנו 9 זרימות:

1. מספר הבתים שעברו בסה"כ בכל הזרימות: 18874368
2. מספר החבילות שעברו בסה"כ בכל הזרימות: 12476
3. קצב הנתונים הכולל, כלומר: כמה בתים הגיעו ליעד בשנייה, בממוצע: 17644893.30
4. קצב החבילות הכולל, כלומר: כמה חבילות הגיעו ליעד בשנייה, בממוצע: 11663.31

```
Please enter the number of streams you want: 9
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Data for stream 5 saved to stream_data_5.bin
Data for stream 6 saved to stream_data_6.bin
Data for stream 7 saved to stream_data_7.bin
Data for stream 8 saved to stream_data_8.bin
Stream 8: Completed sending stream_data_8.bin
Stream 3: Completed sending stream_data_3.bin
Stream 7: Completed sending stream_data_7.bin
Stream 6: Completed sending stream_data_6.bin
Stream 4: Completed sending stream_data_4.bin
Stream 2: Completed sending stream_data_2.bin
Stream 1: Completed sending stream_data_1.bin
Stream 0: Completed sending stream_data_0.bin
Stream 5: Completed sending stream_data_5.bin

Stream Statistics:
```

```
  Data rate: 18348784.42 bytes/second
  Packet rate: 13884.14 packets/second
Stream 6:
  Bytes sent: 2097152
  Packets sent: 1225
  Data rate: 17508356.04 bytes/second
  Packet rate: 10227.08 packets/second
Stream 7:
  Bytes sent: 2097152
  Packets sent: 1057
  Data rate: 19339255.73 bytes/second
  Packet rate: 9747.31 packets/second
Stream 8:
  Bytes sent: 2097152
  Packets sent: 1075
  Data rate: 20664986.94 bytes/second
  Packet rate: 10592.87 packets/second

Overall Statistics:
Total bytes sent: 18874368
Total packets sent: 12476
Average data rate: 17644893.30 bytes/second
Average packet rate: 11663.31 packets/second
```



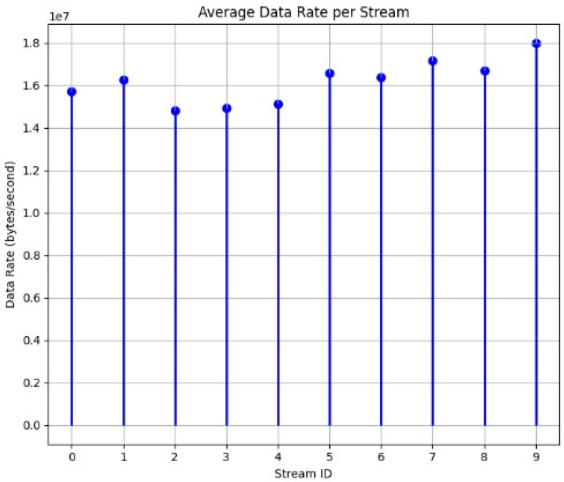



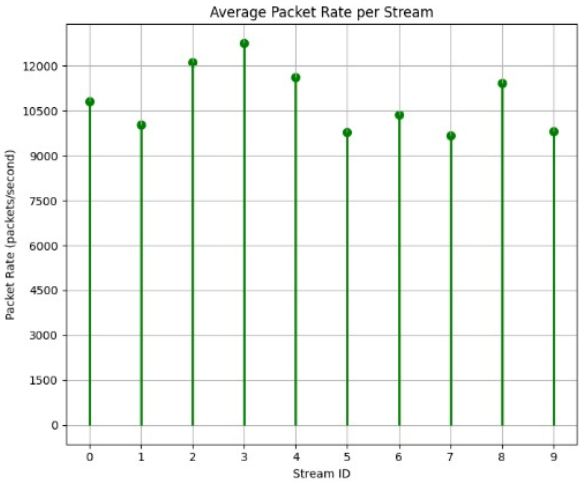
הרצנו 10 זרימות:

1. מספר הבתים שעברו בסה"כ בכל הזרימות: 20971520
2. מספר החבילות שעברו בסה"כ בכל הזרימות: 14185
3. קצב הנתונים הכולל, כלומר: כמה בתים הגיעו ליעד בשנייה, בממוצע: 16107945.21
4. קצב החבילות הכולל, כלומר: כמה חבילות הגיעו ליעד בשנייה, בממוצע: 10895.31

```
Please enter the number of streams you want: 10
Data for stream 0 saved to stream_data_0.bin
Data for stream 1 saved to stream_data_1.bin
Data for stream 2 saved to stream_data_2.bin
Data for stream 3 saved to stream_data_3.bin
Data for stream 4 saved to stream_data_4.bin
Data for stream 5 saved to stream_data_5.bin
Data for stream 6 saved to stream_data_6.bin
Data for stream 7 saved to stream_data_7.bin
Data for stream 8 saved to stream_data_8.bin
Data for stream 9 saved to stream_data_9.bin
Stream 7: Completed sending stream_data_7.bin
Stream 9: Completed sending stream_data_9.bin
Stream 5: Completed sending stream_data_5.bin
Stream 1: Completed sending stream_data_1.bin
Stream 6: Completed sending stream_data_6.bin
Stream 0: Completed sending stream_data_0.bin
Stream 8: Completed sending stream_data_8.bin
Stream 4: Completed sending stream_data_4.bin
Stream 2: Completed sending stream_data_2.bin
Stream 3: Completed sending stream_data_3.bin
```

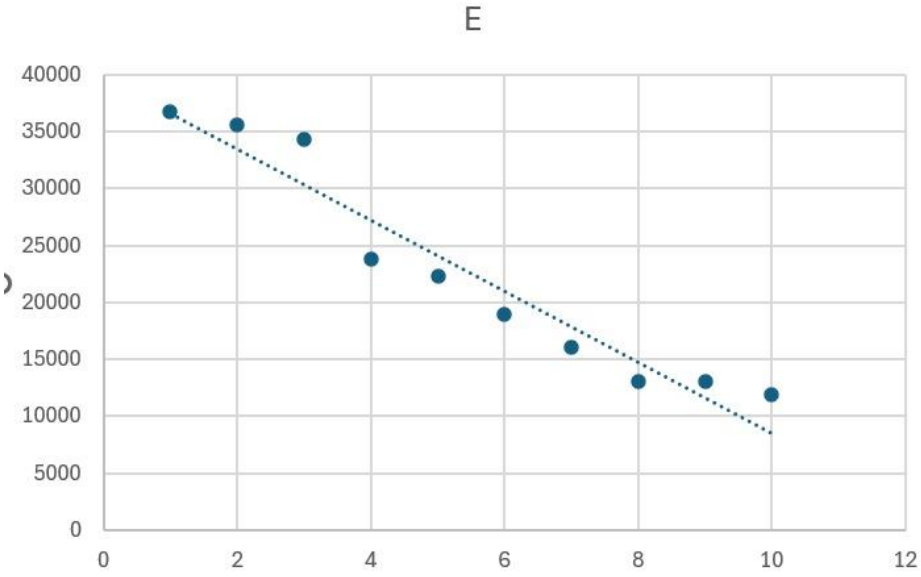
```
Stream 7:
  Bytes sent: 2097152
  Packets sent: 1184
  Data rate: 17152030.63 bytes/second
  Packet rate: 9683.61 packets/second
Stream 8:
  Bytes sent: 2097152
  Packets sent: 1435
  Data rate: 16705967.86 bytes/second
  Packet rate: 11431.25 packets/second
Stream 9:
  Bytes sent: 2097152
  Packets sent: 1145
  Data rate: 17979059.45 bytes/second
  Packet rate: 9816.18 packets/second

Overall Statistics:
Total bytes sent: 20971520
Total packets sent: 14185
Average data rate: 16107945.21 bytes/second
Average packet rate: 10895.31 packets/second
```



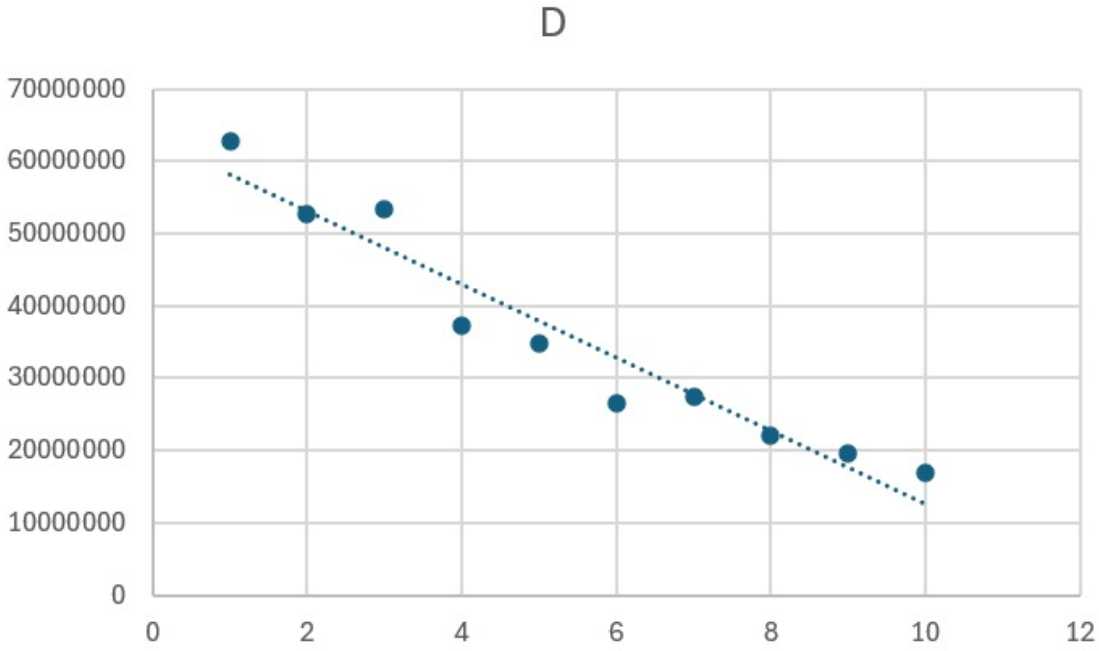
## גרף E

בנינו את הגרף ע״י נקודות שאספנו בכל אחד מהניסויים. כאשר כל נקודה מציינת את מספר החבילות שעברו בסה״כ בכל זרימה




## גרף D

בנינו את הגרף ע״י נקודות שאספנו בכל אחד מהניסויים. כאשר כל נקודה מציינת את מספר הבתים שעברו בסה״כ בכל זרימה.



כאשר מגדילים את מספר הזרימות (flows) ב-QUIC, קצב הנתונים הכולל יורד. הסיבה לכך היא שכאשר מספר הזרימות עולה, יותר נתונים נשלחים במקביל על גבי אותו רוחב פס. אם הרשת או השרתים אינם מסוגלים לטפל בעומס המוגבר, עלולים להיווצר צוואר בקבוק שמשפיעים על היכולת של המערכת להעביר נתונים בקצב גבוה. עומס זה יכול לנבוע ממספר גורמים :

1. **רוחב פס מוגבל:** רוחב הפס הכולל שיש למערכת יכול להיות מוגבל, ואם כל זרימה צורכת חלק מהמשאבים הללו, בסופו של דבר, אין מספיק רוחב פס פנוי לכל הזרימות, והדבר גורם לירידה בקצב הנתונים הכולל.
2. **עומס על השרת:** כאשר מספר זרימות גדול יותר מטיל עומס כבד יותר על השרת, ייתכן שהשרת לא יוכל לעבד את כל הזרימות בזמן אמת, מה שיגרום לעיכובים ולהפחתה בקצב הנתונים הכולל.
3. **עיכובים ברשת:** רשת עמוסה יכולה לגרום לעיכובים, איבוד חבילות, ותקורה נוספת המפחיתה את היעילות הכללית של העברת הנתונים.

**קצב החבילות הכולל:(e)**

כאשר מגדילים את מספר הזרימות (flows) ב-QUIC, קצב החבילות הכולל יורד. ככל שיש יותר זרימות, יש צורך להעביר יותר חבילות במקביל, דבר שיכול להכביד על השרתים והרשת באופן דומה למתואר קודם לכן :

1. **עיבוד חבילות בשרת:** כאשר מספר החבילות שצריך לעבד במקביל גדל, השרת עלול למצוא עצמו מתמודד עם צוואר בקבוק בעיבוד המידע. אם השרת לא מספיק חזק כדי לטפל בכל החבילות בזמן אמת, קצב החבילות הכולל ירד.
2. **תקורה של רשת:** עם מספר גדול יותר של חבילות, יש גם יותר תקורה (overhead) של רשת. תקורה זו יכולה לנבוע מהצורך לאמת חבילות, לבצע תהליך של תיקון שגיאות, ולוודא שהחבילות מגיעות בסדר הנכון.

## בדיקות (Test)

**מטרת הקובץ:**

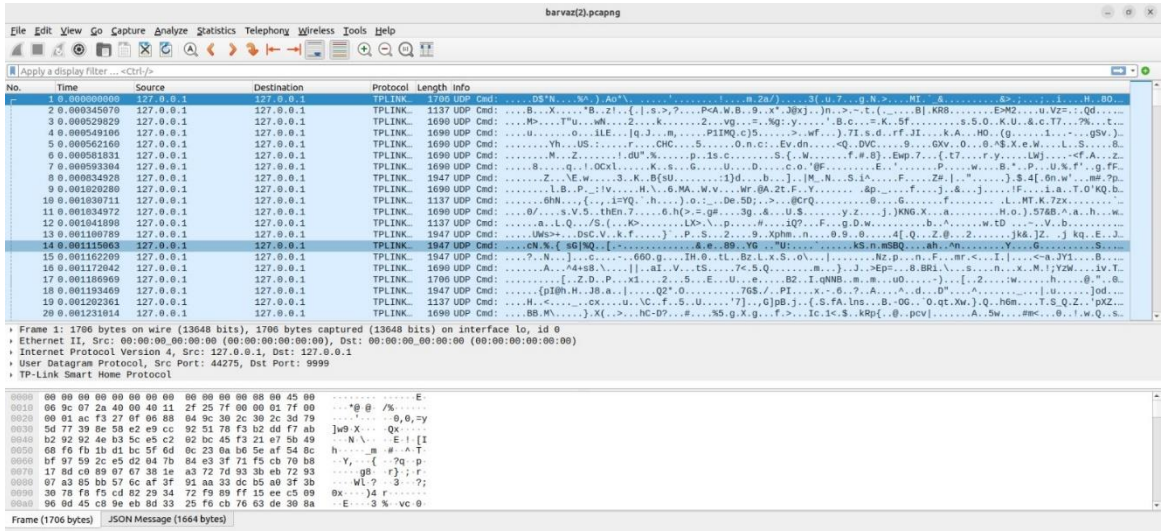
- לבצע בדיקות על מנת לוודא שהנתונים נשלחים ומתקבלים נכון.

**פונקציות עיקריות:**

- runtest(n) : מריצה את השרת והלקוח עם פרמטרים שונים, בודקת את הנתונים שנשלחו והתקבלו ומשווה בין הגודל של הקבצים המקוריים לקבצים שהתקבלו.
- main() : מנהלת את ביצוע הבדיקות על ידי קבלת פרמטרים מהמשתמש, הרצת הבדיקות ודיווח על התוצאות.

## הקלטת WireShark :

מצורפת הקלטה ובה דוגמא להרצת אחד הניסויים ב-Wireshark.




נשים לב לכמה נקודות:

```
Frame 8: 1161 bytes on wire (9288 bits), 1161 bytes captured (9288 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 42101, Dst Port: 9999
Data (1119 bytes)
```

נשים לב כי אכן אנחנו משתמשים בפרוטוקול UDP להעברת הקבצים. וכן נשים לב כי המחשב שולח את הקובץ לעצמו (כתובת ה-IP תמיד 127.0.0.1).

```
Frame 8: 1161 bytes on wire (9288 bits), 1161 bytes captured (9288 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    User Datagram Protocol, Src Port: 42101, Dst Port: 9999
        Source Port: 42101
        Destination Port: 9999
        Length: 1127
        Checksum: 0x027b [unverified]
        [Checksum Status: Unverified]
        [Stream index: 0]
        [Timestamps]
        UDP payload (1119 bytes)
        Data (1119 bytes)
1cc2305a35cd8dc3f3b2bc37fa817eb40218c2d69d9171f095563a923df8372bd9c11192d19c9af8ac6ea362f3ad10dbd67dd369bcd01
        [Length: 1119]
```

כמו כן כאשר נסתכל בפרטי ה-UDP נראה כי ה-Destination Port אכן 9999 כפי שהגדרנו, ועבור כל פקטה יש Stream ID כמו כן כפי שהגדרנו.