בס״ד

פרטי המגישים:
גל בן עמי 206374498
גידי רבי 325483444
אלרואי כרמל 208762971
אהרון בסוס 206751976

קישור לgithub של הפרויקט:

https://github.com/benami171/networks_final_project.git

תיאור כללי של הפרויקט:

הפרויקט שלנו מממש גרסה בסיסית של פרוטוקול QUIC. פרוטוקול QUIC הוא פרוטוקול מודרני הרץ על גבי פרוטוקול UDP, עקרונות פרוטוקול QUIC מספקים מהירות, אמינות ותקשורת מאובטחת.
הפרויקט שלנו מממש:
1. יצירת חיבור בין שני הקצוות
2. שליחת מידע על ידי Streams.
3. ניהול ריבוי זרימות של מידע, הזורמים במקביל.

נתאר את ליבת התוכנית:

1. **הקמת חיבור:**
התקשורת מתבצעת ע״י פרוטוקל udp, הידוע כפרוטוקול מהיר אך לא אמין. לכן ע״מ לבסס אמינות הפרויקט שלנו ממש את עקרון ״לחיצת ידיים משולשת״ שמקורו בפרוטוקול tcp.
מימוש: בתחילת יצירת החיבור ה sender שולח SYN packet ל receiver. לאחר מכן ה receiver עונה ע״ שליחת SYN-ACK בחזרה.

2. **העברת הנתונים:**
לאחר שהחיבור מבוסס, המידע נשלח במספר זרמים.
פרוטוקול QUIC ידוע ביכולתו לממש ריבוי זרימות כלומר מידע רב (בלתי תלוי) יכול להישלח בזרמים שונים במקביל וכך לפתור את בעיית ״Head of line blocking".

כאשר ה Sender שולח את המידע הוא מחלק את המידע ל frames ועוטף אותם ב Packets.
כל Packet נשלחת ב Stearm אחר. כך מובטח שאם יאבד מידע ברשת בstream מסויים שאר ה Packets שזורמות בstreams אחרים ימשיכו להישלח כרגיל.

3. **ניהול הזרמים:**
הפרויקט תומך בריבוי זרמים, כל אחד מזוהה על ידי מזהה זרם ייחודי.
זה מאפשר זרימות נתונים מקבילות, שבהן השולח יכול לשלוח נתחי נתונים שונים בו זמנית, והמקבל יכול לנהל את הזרמים הללו ללא הפרעה ביניהם.

ניהול הזרמים כולל גם מעקב אחר סטטיסטיקות על כמות הנתונים שנשלחו והתקבלו סה״כ , ומספק תובנות לגבי ביצועי החיבור.

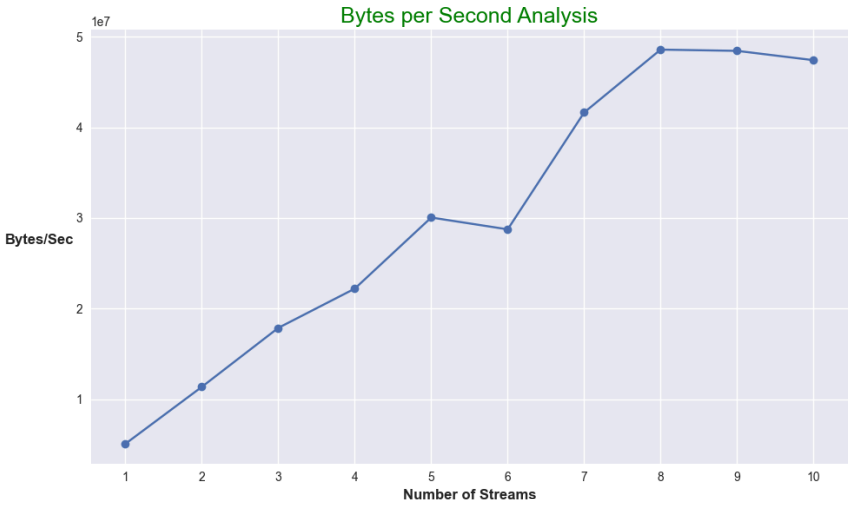
אנחנו בחרנו לממש את עקרון ״ streams multiplex ״ ע״י פעולות אסינכרוניות.

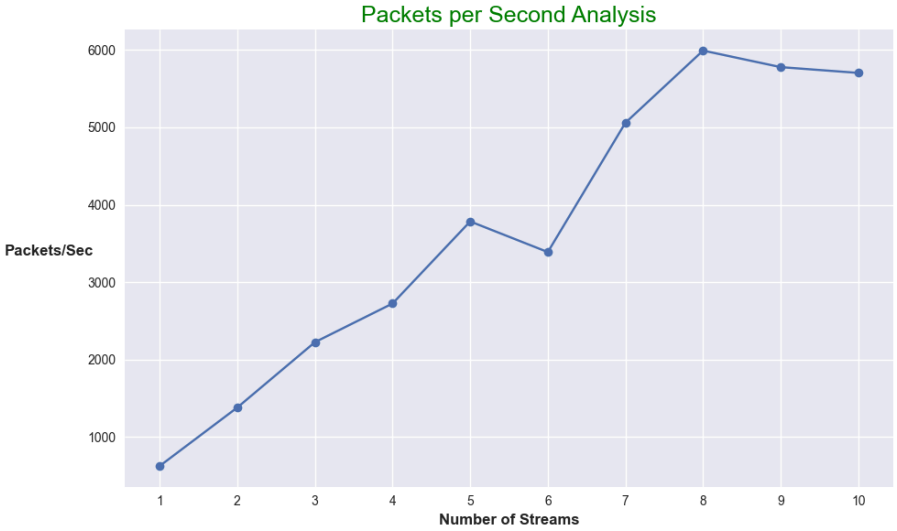
פעולות אסינכרוניות:
גם השולח וגם המקבל משתמשים בתכנות אסינכרוני (דרך ספריית asyncio של Python). יכולת זו מאפשרת למערכת לטפל במספר משימות במקביל מבלי לחסום את ביצועי התוכנית,
יכולת זאת חיוני והכרחית לפרוטוקולי תקשורת בזמן אמת כמו QUIC.

ע״י כך השולח והמקבל מבטיחים שכל הנתונים מועברים ומתקבלים בהצלחה, גם לנוכח אובדן packets פוטנציאלי.

מחקר הביצועים:

ייצרנו קובץ רנדומלי בגודל 1MB. בכל זרם שלחנו העתק של הקובץ. מס׳ הזרמים לאורך הניסוי רץ מ-1 עד 10. נציג גרפים של קצב הנתונים הכולל עבור כמות הבתים לשנייה בממוצע וכמות חבילות לשנייה בממוצע:





ניתן לראות שהגרף במגמת עלייה וככל שמגדילים את מס׳ הזרימות קצב הנתונים הכולל עולה, מכיוון שכמות הנתונים הנשלחת עולה, ומכיוון שאנו שולחים מידע **במקביל** ע״י ריבוי זרימות זמן השליחה גדל משמעותית בקצב קטן יותר לכן קצב הנתונים הכולל יגדל.

תשובות לשאלות על המאמר:

1. <u>5 חסרונות של TCP:</u>

- Head of Line Blocking:

בTCP קיים מנגנון שמוודא שהחבילות שיועברו לשכבת האפליקציה יהיו לפי הסדר הנכון, כלומר לפי סדר שליחתם ע״י השולח. כתוצאה מכך יכול להתקיים מצב שבו החבילה הראשונה שנשלחה נאבדה בדרך אך לדוג׳ חמשת החבילות הבאות כן הגיעו, אך בגלל שהראשונה לא הגיעה, שכבת התעבורה תשאיר אצלה את החבילות עד שהיא תקבל את החבילה הראשונה ורק אז תעביר את כל החבילות לשכבת האפליקציה. מצב זה לא יעיל מהבחינה שכל עוד החבילה הראשונה לא הגיעה שאר החבילות סתם תופסות מקום בתור אצל שכבת התעבורה ולא מאפשרות לשאר החבילות להמשיך להגיע.

- רגישות לכתובת IP:

ב-TCP הזהות הייחודית של החיבור מבוססת על הזוג הסדור של (כתובת IP, מספר PORT) של כל אחד ממשתמשי הקצה, לכן כל שינוי מבחינה זו אצל כל אחד מהם גורם לסגירת החיבור ביניהם והתחלת יצירת החיבור מחדש כך שכל המידע שהועבר ע״י החיבור הנוכחי ירד לטמיון וכך יוצא שעל מנת ליצור קשר מחדש, הלקוח והשרת צריכים ליצור לחיצת ידיים חדשה.

- שיהוי גדול בתחילת חיבור:

חיבור TCP נוצר ע״י לחיצת ידיים משולשת, מה שעלול ליצור איטיות בקבלת המידע המבוקש מהצד השני. במקרה שנעשה שימוש בTLS אז נוצר עוד יותר עומס מכיוון שזה מוסיף שלבים בתחילת ההתקשרות, לתהליך הTLS יש תהליך לחיצת ידיים משלו בו הלקוח שולח הודעה לשרת עם הגדרות הצפנה מסויימות להתקשרות ביניהם (כמו באילו גרסאות TLS הוא תומך, "מפתח" רנדומלי המורכב מ32 בתים, וכו'), השרת בוחר אילו הגדרות הוא רוצה ושולח אישור ללקוח כדי לקבל אימות ואז הלקוח והשרת מחליפים ביניהם את המפתח ומאשרים שקיבלו.
כמו שאמרנו, התהליך הזה מעכב את תחילת שליחת המידע.

- גודל header קבוע:

ב-header של TCP קיימים שדות שגודלם הקבוע השפיע על האפקטיביות של פרוטוקול TCP ככל שמהירות האינטרנט הלכה וגדלה. מצד אחד, שדות ה-SEQ, ACK בגודל של 4 בתים. מכיוון שמהירות הרשת גדלה 4 השדות הללו מגיעים מהר מאוד לגודל המקסימלי שלהם ולכן מתאפסות ומתחילות מחדש וכך נוצר מצב שהערכים שלהם כבר לא ממש ייחודיים. מצד שני, שדה ה-WINDOW-SIZE רק בגודל 2 בתים, מה שמגביל את התפוקה של החיבור.

- ערבוב בין בקרת עומס לשליחה אמינה:

בTCP המנגנון שקובע כמה דאטה אפשר לשלוח (congestion control) והמנגנון שמוודא שכל הדאטה אכן הגיעה (ז"א מאפשר reliable delivery) מחוברים יחד באופן הזה שהם משתמשים

בחלון השליחה.

הבעיה היא במקרים כמו המקרה הבא: הלקוח מחזיק חלון שליחה שמשקף את המצב בתעבורת הרשת, ז"א נניח שחלון השליחה הוא בגודל של 8, אז שמונת החבילות שבתוך החלון הן בגדר חבילות שנמצאות כרגע ברשת ועדיין לא קיבלנו אימות שהן אכן הגיעו,נניח שחבילה מס' 2 בחלון לא התקבלה אבל שאר החבילות כן, במקרה כזה על הCC לעצור את התקדמות החלון עד שחבילה מס' 2 תגיע ונקבל עליה אימות. בזמן ההמתנה לשליחה מחדש ואימות על קבלתה של חבילה מס' 2 שנאבדה, חלון השולח לא ישקף את המצב בתעבורת הרשת מכיוון שכרגע אין חבילות ברשת אך לפי חלון השולח יש 7 חבילות שלכאורה נמצאות עדיין ברשת כיוון שהוא לא קיבל עליהן אימות.

2. 5 תפקידים שפרוטוקול תעבורה צריך למלא:

- בקרת עומס: צריך לשלוט על כמות החבילות בתוך הרשת בזמן נתון.
- אבטחה: מנגנון שתפקידו לוודא שהמידע שעובר ברשת יעבור באופן מאובטח ושהתקשורת עצמה בין לקוח לשרת תהיה מאובטחת. למשל פרוטוקול TCP משתמש בTLS להצפנה.
- העברת מידע באופן אמין: ממומש במספר אופנים, למשל חלון בקרת זרימה שנועד למנוע את בעיית ה"head-of-line" שציינו לעיל. בנוסף שימוש במזהה ייחודי למידע - למשל sequence number של חבילות שמאפשר למי שמקבל את המידע לוודא שאכן המידע התקבל בסדר הנכון.
- הגדרת מזהה ייחודי לחיבור ומזהה ייחודי למידע הנשלח
- ניהול מאפייני החיבור. צריך מזהה ייחודי גלובלי לחיבור בין 2 הנקודות קצה, בשאיפה שיהיה בלתי תלוי בכתובות IP שלהם (כך שיתמוך בהמשך החיבור למרות שכתובת IP כלשהי השתנתה), אבל שידע למפות לכתובות IP בצורה אמינה. צריך מנגנון מסודר להקמת החיבור וסגירתו. ובנוסף, שיתמוך בפורמט הודעות לכל אחד מהצדדים בנוגע *למאפייני החיבור עצמו*.

3. אופן פתיחת הקשר (״לחיצת ידיים״) ב-QUIC

תהליך לחיצת הידיים:

- הלקוח שולח חבילה ראשונית שכוללת הודעת פתיחה של TLS עם הודעת ClientHello,איזה אלגוריתמי הצפנה נתמכים על ידו, החלק הציבורי של המפתח שלו, ופרמטרים של פרוטוקול QUIC.
- השרת מגיב עם חבילה ראשונית שמכילה בתוכה הודעת TLS ServerHello, אלגוריתם הצפנה נבחר, ומפתח ציבורי של השרת. בנוסף שולח חבילת לחיצת ידיים עם הרחבות מוצפנות, נתוני אימות של השרת (כמו דומיין ומפתח ציבורי), אישור של נתוני האימות שמוכיח שהשרת אכן מחזיק את המפתח הפרטי שתואם למפתח הציבורי, והודעת TLS finished.
- הלקוח לאחר מכן שולח חבילת לחיצת ידיים עם הודעת TLS Finished
- השרת מאשר עם חבילה של 1 RTT שמכילה את הפריים HANDSHAKE_DONE.

כיצד בא לידי ביטוי השיפור של חלק מהחסרונות של TCP:

- עיכוב הנובע מפתיחת קשר - QUIC משלב את תהליך הTLS ולחיצת הידיים לתוך תהליך של RTT אחד, מה שמוריד את זמן יצירת הקשר בהשוואה ל TCP עם TLS שדורשים בדרך כלל בין 2 ל 3 RTT. בנוסף לכך בעת חידוש קשר פרוטוקול QUIC תומך בזה ב0 RTT מה שעוד יותר מוריד את זמן העיכוב.
- רגישות לכתובת IP - פרוטוקול QUIC משתמש בconnection ID שהוא בלתי תלוי בכתובת IP וזה מאפשר לקשר לשרוד שינויים בהגדרות רשת כמו שינוי כתובת IP בלי לנתק את הקשר בניגוד ל TCP אשר קושר באופן מובהק את החיבור לכתובות הIP והפורט.
- גודל header קבוע - בניגוד לחיבור TCP בחיבור QUIC אנחנו משתמשים בשני סוגים של Headers, אחד קצר ואחד ארוך. הארוך משמש להקמת קשר ומכיל יותר מידע, הקצר משמש להעברת דאטה.
  זה עוזר לנו לטפל בסוגים שונים של חבילות בצורה יעילה, וגם מאפשר להוסיף תכונות חדשות בלי לנתק את הקשר הקיים.
  פרוטוקול QUIC משתמש בגדלים שונים לייצוג של מספרים. עבור מספרים קטנים, QUIC יקצה גודל של בית 1, ועבור מספרים גדולים יקצה 2, 4 או 8 בתים. וכך בעצם ניתן לחסוך מקום כאשר שולחים מספרים קטנים, וגם יש אופציה להשתמש במספרים גדולים מאוד אם יש צורך בניגוד לTCP שמשתמש באותו גודל לייצוג של כל שדה.

- Head of Line Blocking - בחיבור אחד QUIC תומך במספר streams בלתי תלויים בו זמנית, אם נאבדת חבילה זה משפיע אך ורק על הstream עם דאטה בחבילה הספציפית וזה מאפשר לכל הstreams האחרים להמשיך כרגיל, מה שמוריד משמעותית את החסימה של התור בהשוואה ל TCP.
- ערבוב בין בקרת עומס לשליחה אמינה - פרוטוקול QUIC עושה הפרדה בין בקרת העומס לשליחה אמינה, בנוגע לשליחה אמינה הסברנו בחלק של השיפור של "head of line blocking" שquic משתמש בחיבור אחד בכמה streams בלתי תלויים שאפשר לנהל אותם בנפרד בלי להפריע לתהליך ההעברה של המידע בזרמים השונים. בנוסף לכך QUIC בניגוד לTCP לא מבצע הקטנה של החלון התעבורה אלא אם כן הוא מזהה עיכוב עקבי בתהליך העברת המידע למשל איבוד של כמות משמעותית של חבילות ברציפות בלי לקבל שום ack ביניהם.

4. תארו בקצרה את מבנה החבילה של QUIC. כיצד הוא משפר חלק מהחסרונות של TCP שתיארתם בסעיף 1?

ראשית, לחבילות QUIC יש 2 סוגי header. לחבילה הראשונית שמשמשת לפתיחת הקשר יש header ארוך ולשאר החבילות שיישלחו במהלך הקשר יש header קצר. מאפיין זה של QUIC משפר את החיסרון של TCP בנוגע לגודל header קבוע בכך שנותן ל-QUIC גמישות וניצול טוב יותר של מקום.

שדות ה-header הקצר של QUIC הם:

- Flags: דגלים המצביעים על סוג החבילה ועל פרטים נוספים הקשורים לחבילה.
- Destination Connection ID: זהות חיבור היעד. בשלב הקמת הקשר כל צד שולח לשני מה ה-Connection ID שלו, ובהמשך משתמשים בו לדעת למי נשלחה חבילה. שימוש בשדה זה עונה על הבעיה שציינו שיש ב-TCP בנוגע לנדידת חיבור בכך שגם אם כתובת ה-ip וה-port של אחד מהצדדים משתנה, כל עוד הם שומרים על ה-connection ID אפשר להמשיך את השיחה ללא לחיצת יד משולשת וללא איבוד המידע שנשלח עד כה.
- Packet Number: מספר החבילה שנשלחה (מספר מונוטוני עולה)

- Protected Payload: ה-frames שמרכיבים את החבילה. בחבילה אחת יכולים להיות כמה סוגים שונים של frames.

ב-QUIC יחידת המידע הבסיסית היא frame ויש כמה סוגים. ה-frame הבסיסי לשליחת מידע הוא STREAM Frame. השדות ב-frame זה הם:

- Stream ID: מספר הזרם ש-frame זה משוייכת אליו. בניגוד ל-TCP בו כל התקשורת מתבצעת ב"צינור" אחד, ולכן נוצרת בעיית Head of Line Blocking התקשורת ב-QUIC מתבצעת במספר "צינורות" במקביל הנקראים streams. כתוצאה מכך כאשר חבילה אובדת, רק הזרמים עם מסגרות הנתונים הכלולות בחבילה יצטרכו להמתין לשידור חוזר של המסגרות האבודות. זה לא יחסום זרמים אחרים מלהתקדם.

5. מה QUIC עושה כאשר חבילות מגיעות באיחור או לא מגיעות כלל?

כאשר חבילות QUIC מגיעות באיחור או לא מגיעות כלל, הפרוטוקול משתמש במספר מנגנונים לזיהוי ושחזור אובדן חבילות:

- **זיהוי אובדן חבילות מבוסס אישור (ACK-based loss detection):** יQUIC מבצע זיהוי אובדן חבילות על סמך קבלת אישורים. כאשר חבילה מאוחרת יותר מתקבלת ומאושרת, חבילות שנשלחו לפניה ולא אושרו נחשבות כאבודות.
- **סף מספרי (Packet number threshold):** אם מספר הסדר של החבילה שלא אושרה נמוך בהרבה ממספר החבילה המאושרת האחרונה, היא תיחשב כאבודה.
- **סף זמן (Time threshold):** אם חבילה נשלחה לפני פרק זמן מסוים (מבוסס על ה-RTT) ולא התקבלה עליה אישור, היא תיחשב כאבודה.
- **זיהוי אובדן בתור הסיום (Tail loss detection):** יQUIC משתמש בטיימר עבור זיהוי אובדן חבילות בסוף התור. אם הטיימר פג תוקף מבלי שהתקבל אישור, השולח ישלח חבילה חדשה כ"פרוב" (probe) עם נתונים חוזרים כדי לוודא הגעת הנתונים.
- **מנגנון PTO יProbe Timeout):** כאשר הטיימר פג תוקף מבלי שהתקבל אישור, השולח ישלח חבילה חדשה כ"פרוב" (probe), שיכולה להכיל נתונים חוזרים במטרה להפחית את מספר השידורים החוזרים המיותרים.

6. תארו את בקרת העומס של QUIC.

בקרת העומס ב-QUIC מתמקדת בהבטחת יעילות התעבורה ברשת תוך מניעת עומסים מיותרים. להלן התכונות המרכזיות של בקרת העומס ב-QUIC:

- **הפרדת בקרת העומס מבקרת האמינות:** QUIC משתמש במספרי חבילות לשליטה על העומס ובמספרי מסגרות (stream frame offset) לאמינות ההעברה. זאת בניגוד ל-TCP שמשתמש באותו חלון לשני המטרות.
- **שימוש באלגוריתמים קיימים:** QUIC משתמש במנגנון חלון תעבורה מבוסס, בדומה ל-TCP. אין ל-QUIC אלגוריתם חדש משלו לבקרת עומס, והשולח יכול ליישם מנגנוני בקרת עומס שונים לפי הצורך, כמו Cubic.
- **מניעת הפחתת חלון תעבורה מיותרת:** QUIC לא מצמצם את חלון התעבורה אלא אם כן מזהה עומס מתמשך. עומס מתמשך מזוהה כאשר שתי חבילות רצופות הדורשות אישור לא אושרו, והזמן בין שליחתן עולה על משך זמן מוגדר (מבוסס על ה-RTT הממוצע והסטייה שלו).
- **שליחת נתונים בקצב מתון (Pacing):** השולח יבטיח שהמרווח בין שליחת חבילות יעלה על סף מסוים, שנקבע לפי זמן העברה ממוצע של הרשת (RTT), גודל חלון התעבורה וגודל החבילה. זה נועד למנוע יצירת עומס זמני ברשת.

- **השתמשות באלגוריתם NewReno:** יQUIC משתמש במנגנון בקרת העומס NewReno, הכולל שלושה מצבים: התחלה איטית (Slow Start), התאוששות (Recovery) ומניעת עומס (Congestion Avoidance). במצב של התחלה איטית, חלון התעבורה גדל באופן אקספוננציאלי עד לאיתור אובדן חבילה. במצב התאוששות, חלון התעבורה מצומצם בחצי והמעבר למניעת עומס מתבצע כאשר מתקבל אישור עבור חבילה שנשלחה במצב התאוששות. במצב מניעת עומס, חלון התעבורה גדל באופן איטי יותר, לפי גודל חבילה אחת עבור כל אישור.

בקרת העומס ב-QUIC מתמקדת בשמירה על יעילות התעבורה תוך שמירה על אמינות ההעברה ומניעת עומסים מיותרים, וזאת באמצעות שימוש במנגנונים מתקדמים לזיהוי אובדן ושחזור חבילות.