פרוייקט סיום QUIC

סטודנטים:

- משה אסקרוב 314085986

- אברהים מג'אדלה 207101734

-עבדאללה זועבי 211407424

-אהרון נתן פרטוש 342633106

האפשרות שבחרנו למימוש של QUIC בחלק ה"רטוב":

(1) ריבוי זרימות(FLOWS)

הסבר על הקוד:

api.py : מכיל פונקציות שימושיות גם ל-SERVER וגם ל-CLIENT.

-הסבר קצר על הפונקציות:

generate_connection_id() מחזירה מספר רנדומאלי בין 1-1000000 שמשמש כמספר מזהה של חיבור שנוצר.

generate_packet_number() מחזירה מספר רנדומאלי בין 1-1000000 שמשמש כמספר מזהה של חבילה.

pack_long_header(connection_id, packet_number, flow_id, payload) מקבלת מזהה חיבור, מזהה חבילה ,מזהה זרם ,מטען (המידע שרוצים להעביר) ומחברת אותם לחבילה אחת.

pack_short_header(packet_number , flow_id, payload) מקבלת מזהה חבילה , מזהה זרם ,מטען ומחברת אותם לחבילה אחת.

unpack_long_header(data) מקבלת חבילה ומחלצת ממנה את מזהה החיבור, מזהה זרם , מספר החבילה והמטען.

unpack_short_header(data) מקבלת חבילה ומחלצת ממנה את מספר החבילה, מזהה זרם והמטען.

generate_random_file(size) יוצרת קובץ(מטען) רנדומאלי בגודל שמתקבל.

print_statistics(flows) מקבלת מערך של מילונים(flow) שכל מילון מכיל מספר מזהה של אותו מילון, גודל חבילה, מספר ביטים, מספר חבילות, זמן התחלה ,זמן סיום ומדפיסה את נתוני הזרימות.

client.py :

start_quic_client() מבצעת את הפעולות הבאות:

1.יוצרת client socket(UDP)

2. אורזת חבילה ראשונית עם כותרת ערוכה ושולח לשרת.

3.מקבלת מהשרת חבילה עם כותרת קצרה ומחלצת ממנה מידע.
אם החבילה היא חבילת סגירת חיבור אז היא שולחת בחזרה הודעת סגירה וסוגרת את החיבור מהצד של הלקוח.

אם החבילה היא חבילת חיבור ראשונית אז היא מדפיסה שהתקבלה חבילה ומחכה לחבילה הבאה.

אם החבילה היא לא חבילת סגירת חיבור ולא חבילת חיבור ראשונית אז היא מחלצת את המידע שהגיע ומעדכנת נתונים.

4. היא חוזרת על שלב 3 עד שכל הזרמים הושלמו, ברגע שהזרימות הושלמו היא מקבלת הודעת סגירת חיבור מהשרת וסוגרת.

5. ברגע שכל הזרימות הושלמו עם print_statistics היא מדפיסה את הנתונים ובעזרת show_graph מציגה גרף שמראה ממוצע העברת מידע וממוצע שליחת חבילה לפי מספר הזרימות.

send_connection_close(client_socket, server_address)

פונקציה זו החראית לשלוח הודעת סגירת חיבור (Connection Close) לשרת, שאומרת לשרת שהלקוח מבקש לסיים את החיבור.

הפונקציה מקבלת כתובת שרת ,socket, אורזת חבילה של סגירת חיבור ושולחת לשרת.

show_graph(flows)

פונקציה זו החראית להדפיס גרף שמציג לנו ממוצע העברת מידע וממוצע שליחת חבילה.
הפונקציה מקבלת רשימה של מילונים מחלצת מהם את כמות הזרימות ממוצע העברת מידע ,ממוצע שליחת חבילה ומדפיסה גרם שמציג את הנתונים.

server.py :

start_quic_server() מבצעת את הפעולות הבאות:

1. יוצר מילון(connections) שיהיה בו מידע על חיבורים קיימים כגון רשימת הזרימות ומזהה חיבור של הלקוח.

2.יוצר server socket(UDP)

3.מאזין ומחכה לחבילה מהלקוח.

4. אם החבילה שמתקבלת מגיע מלקוח שלא שמור בconnections אז הוא מתייחס אליה כחבילה עם כותרת ערוכה ,מאתחל את הנתונים של הלקוח בconnections , שולח חבילת חיבור ראשונית בעזרת send_client_hello, מאתחל את הזרימות.

לאחר מכן היא שולחת את החבילות ללקוח בעזרת send_files_to_client וסוגרת חיבור בעזרת send_connection_close.

5.אם החבילה מגיעה מלקוח שכן שמור בconnections אז היא בודקת אם זה חבילת ACK שבמידה וכן היא מעדכנת את acks בconnections או חבילת סגירת חיבור שמובילה לסגירת החיבור..

get_num_flows()

פונקציה זו מחזירה את מספר הזרימות שמוגר בתחילת הקוד.

send_client_hello(server_socket, client_address, connection_id)

החראית על יצרת חבילת חיבור ראשונית ולשלוח אותה ללקוח, היא מקבלת socket כתובת לקוח ומזהה חיבור אורזת חבילת חיבור ראשונית עם מטען CLIENT_HELLO שמוגדר מראש ושולחת ללקוח.

send_files_to_client(server_socket, client_address)

פונקציה זו מחלצת את רשימת הזרימות לפי הנתונים בconnections ומשתמשת ב send_next_packet בכדי לשלוח את המידע בכל זרם.

send_next_packet(server_socket, client_address, flow)

פונקציה זו שולחת את החבילה הבאה בזרימה מסוימת ללקוח. היא בודקת אם יש עוד נתונים לשלוח בזרימה, מעדכנת מספר חבילה, אורזת את הנתונים עם כותרת קצרה ,שולחת את החבילה ללקוח ומעדכנת את מספר הבתים והחבילות שנשלחו בזרימה זו.

send_connection_close(server_socket, client_address)

החראית על שליחת הודעת סגירת חיבור (CONNECTION_CLOSE) ללקוח, על מנת לסיים את החיבור. היא אורזת את ההודעה(שמוגדרת מראש בapi.py) עם כותרת קצרה ושולחת את החבילה ללקוח. לאחר מכן, היא ממתינה קצת וסוגרת את השרת.

test.py:

TestAPI(unittest.TestCase)

מחלקה שהחראית על בדיקה של כל הפונקציות בapi.py.

הבדיקות חוזרות על עצמן 50 פעם.

test_generate_connection_id(self) קוראת לפונקציה generate_connection_id ובודקת שהמספר שחוזר הוא בין 1 ל 1000000 כולל.

test_generate_packet_number(self) קוראת לפונקציה generate_packet_number ובודקת שהמספר שחוזר הוא בין 1 ל 1000000 כולל.

test_pack_and_unpack_long_header(self) נוצרים מזהה חיבור, מזהה חבילה ומטען רנדומליים אותם אורזים לחבילה בעזרת pack_long_header ואחר כך מפרקים אותם בחזרה בעזרת unpack_long_header ובודקים שהערכים נשמרים.

test_pack_and_unpack_short_header(self) נוצרים מזהה חבילה ומטען רנדומליים אותם אורזים לחבילה בעזרת pack_short_header ואחר כך מפרקים אותם בחזרה בעזרת unpack_short_header ובודקים שהערכים נשמרים.

test_generate_random_file(self) משתמשת בפונקציה generate_random_file בכדי ליצור קובץ בגודל רנדומאלי ובודקת שהקובץ שהתקבל הוא בגודל הרצוי וכולל בתוכו ביטים

TestClientServer(unittest.TestCase)

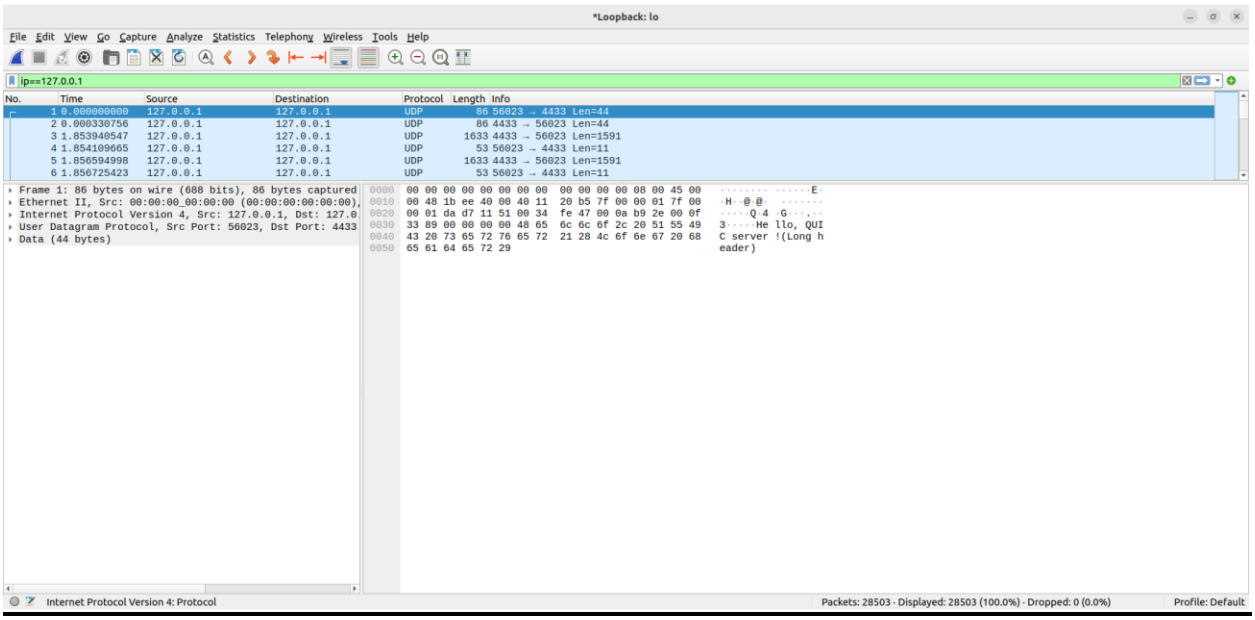
Threading: בעזרת "Threading " נאכל לבצע בדיקה שבא גם הסרבר וגם המשתמש רצים "באותו הזמן" ונאכל לדמא מקרה של חיבור ותקשורת בין המשתמש לשרת

start_quic_server_thread(self) היא פונקציית עזר ל test_quic_client_server_comm והיא פותחת "thread" שמפעיל את start_quic_server ומשמש אותנו כהסרבר

test_quic_client_server_comm(self) היא הפונקציה המרכזית שמשתמשת בפונקציית העזר start_quic_server_thread בכדי לפתוח thread של הסרבר, לאחר מכן היא פותחת thread שמשמש אותנו כהמשתמש. כששני הthreads פתוחים מתבצעת התקשורת בינתיים כפי שמצויין בקוד של server.py וclient.py ואם הכל מתבצע ללא שגיעות הבדיקה עברה בהצלחה

הסברי ווירשארק:

(0.1)



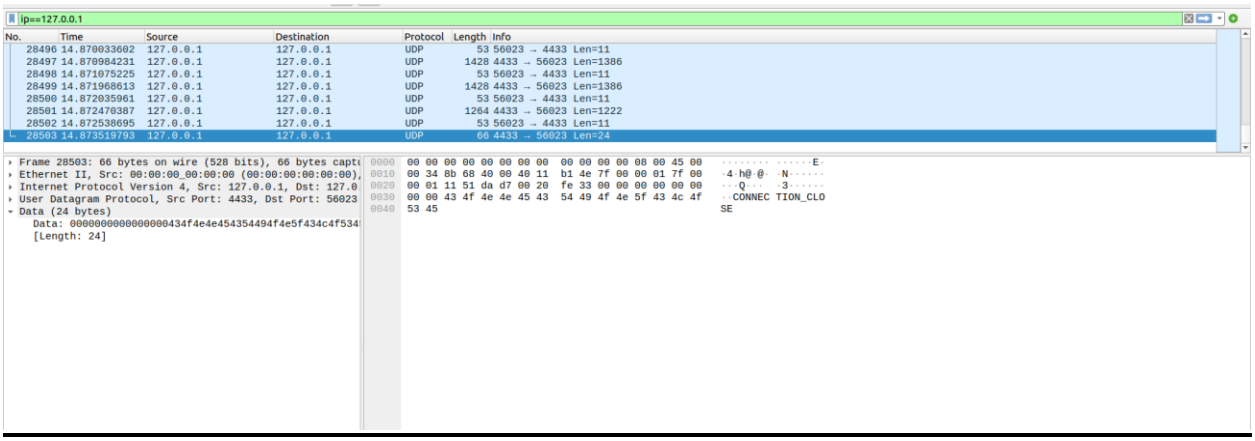
פה רואים את חבילת פתיחת הקשר (long header) מצד הלקוח לשרת ומספרה 1 וגודלה 89 ביטים. כאשר מכילה את המידע connection ID, packetnumber, flow id והשרת מבצע עליה unpack.

חבילה מספר 2 היא התגובה מהשרת וגם long header על קבלת חבילה פתיחת קשר.

חבילה מספר 3 חבילת ה data הראשונה כאשר הייתה בחירה באופן אקראי עבור גודל כל חבילה בזרימה הראשונה והגודל שהוגרל היה 1633 ביטים וסה"כ בכל זרימה MB2 תמיד, ונשלחת ב short header שמכיל packet number, flow id, connection id, data, ולקוח עושה unpack.

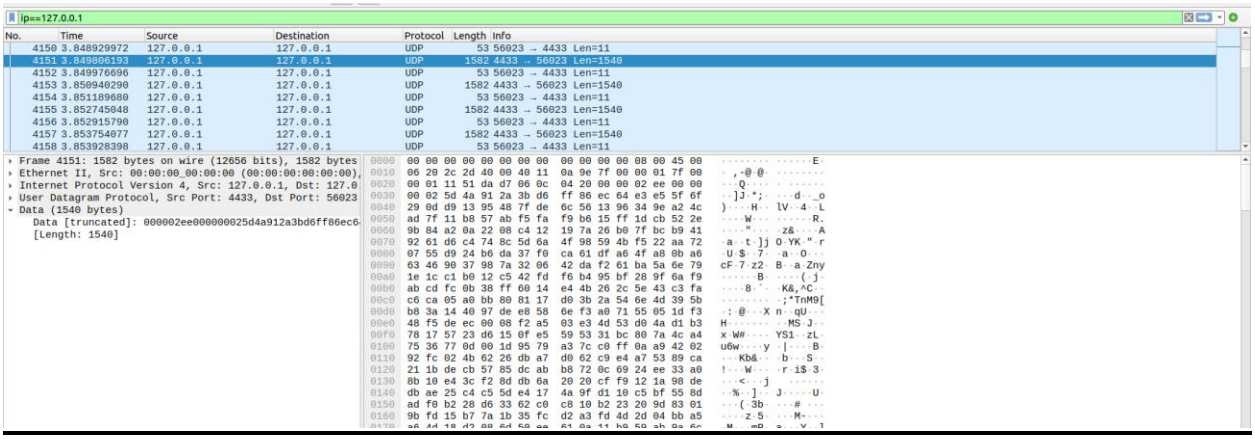
ואחרי שליחתה יש חבילת ack מוחזרת באופן סדרתי מלקוח לשרת אחרי כל שליחה לחבילה והוא בגודל 53 ביטים סה"כ.

(0.2)



עכשיו אנו מסתכלים על זרימה אחרת כאשר גודל החבילה שהוגרל כאן הוא 1582 ביטים, ונשלחת ב short header גם. והלקוח מגיב על קבלתה בהצלחה (ואותו הדבר לכל הזרימות האחרות).

(0.3)



כאן ואחרי שליחת כל הזרימות וקבלת ה acks עבורם, נשלחת הודעת Connection_close רואים שהיא האחרונה ומספרה 28503 כדי להודיע ללקוח על סיום שליחת כל הזרימות וסגירת הקשר. הלקוח מעבד את הבקשה ומתכונן לסגירה.

הפלט והגרפים עבור מספרי זרימות שונים

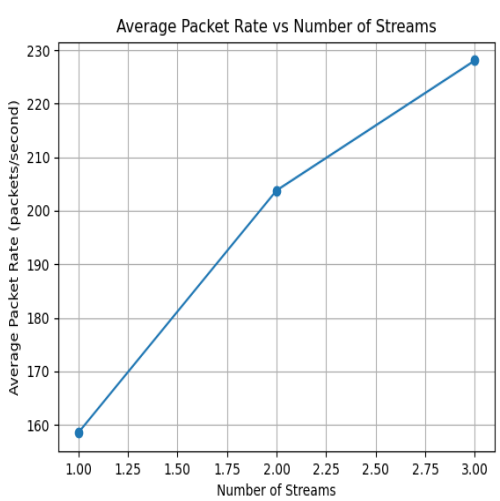
הערות:

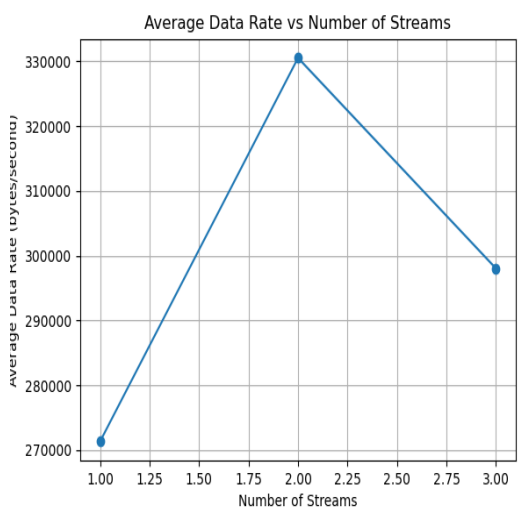
1. בכדי לשנות את כמות הזרימות ניתן לבצע שינוי למשתנה num_flows ב server.py.

2. השתמשנו ב sleep כאשר הוא סייע לעזור במקרה של איבוד החבילות שקיבלנו מרוב מספר החבילות, ועבור ערכי איחור שונים לפונקציית sleep שונים קיבלנו פלטים חלקם יעילים ולכן תיאמנו את הזמן למקרים הכי טובים שקיבלנו.

3-זרימות

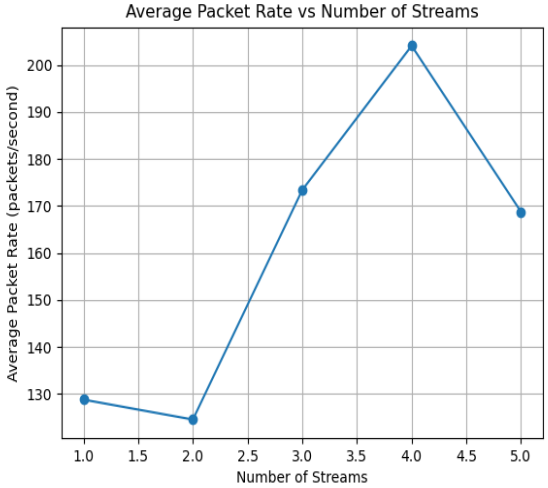
```
Initialized flow 1
Set packet size for flow 1 to 1713 bytes.
Flow 1 is complete
Initialized flow 2
Set packet size for flow 2 to 1622 bytes.
Flow 2 is complete
Initialized flow 3
Set packet size for flow 3 to 1307 bytes.
Flow 3 is complete
Flow statistics:
Flow 1:
  Total bytes: 2097152
  Total packets: 1225
 Packet Size is : 1713 bytes
  Data rate: 271309.99 bytes/second
  Packet rate: 158.48 packets/second
Flow 2:
  Total bytes: 2097152
  Total packets: 1293
 Packet Size is : 1622 bytes
  Data rate: 330536.47 bytes/second
  Packet rate: 203.79 packets/second
Flow 3:
  Total bytes: 2097152
  Total packets: 1605
 Packet Size is : 1307 bytes
  Data rate: 298029.24 bytes/second
  Packet rate: 228.09 packets/second
Overall statistics:
  Average data rate: 299958.56 bytes/second
  Average packet rate: 196.79 packets/second
abode@abode-virtual-machine:~/Downloads/pythonProject$
```

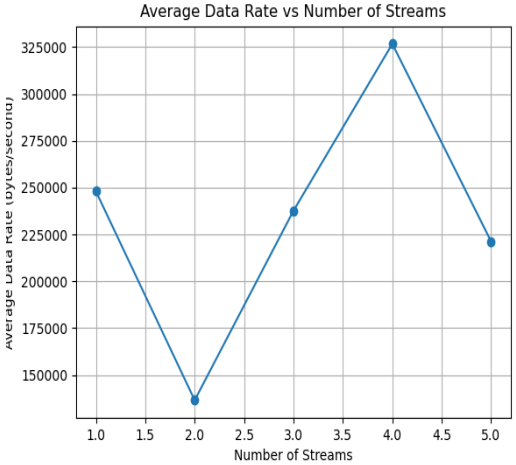




5-זרימות

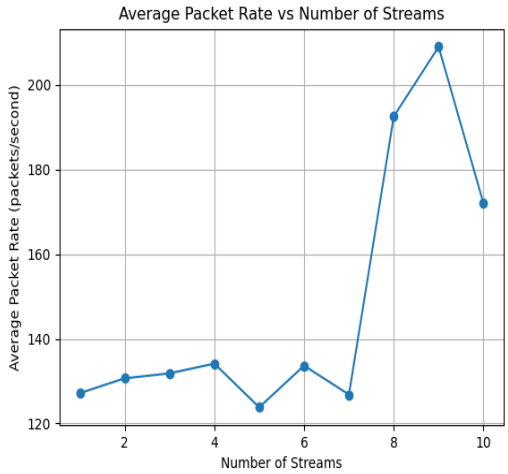
```
Initialized flow 4
Set packet size for flow 4 to 1602 bytes.
Flow 4 is complete
Initialized flow 5
Set packet size for flow 5 to 1311 bytes.
Flow 5 is complete
Flow statistics:
Flow 1:
  Total bytes: 2097152
  Total packets: 1089
 Packet Size is : 1926 bytes
  Data rate: 248033.67 bytes/second
  Packet rate: 128.80 packets/second
Flow 2:
  Total bytes: 2097152
  Total packets: 1914
 Packet Size is : 1096 bytes
  Data rate: 136468.82 bytes/second
  Packet rate: 124.55 packets/second
Flow 3:
  Total bytes: 2097152
  Total packets: 1530
 Packet Size is : 1371 bytes
  Data rate: 237692.17 bytes/second
  Packet rate: 173.41 packets/second
Flow 4:
  Total bytes: 2097152
  Total packets: 1310
 Packet Size is : 1602 bytes
  Data rate: 326751.49 bytes/second
  Packet rate: 204.11 packets/second
Flow 5:
  Total bytes: 2097152
  Total packets: 1600
 Packet Size is : 1311 bytes
  Data rate: 221199.59 bytes/second
  Packet rate: 168.76 packets/second
Overall statistics:
  Average data rate: 234029.15 bytes/second
  Average packet rate: 159.93 packets/second
abode@abode-virtual-machine:~/Downloads/pythonProject$
```

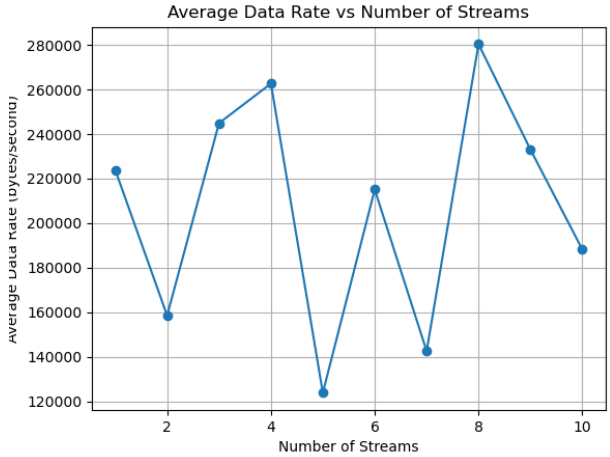




10-זרימות

```
    Packet rate: 134.13 packets/second
  Flow 5:
    Total bytes: 2097152
    Total packets: 2096
   Packet Size is : 1001 bytes
    Data rate: 123887.94 bytes/second
    Packet rate: 123.82 packets/second
  Flow 6:
    Total bytes: 2097152
    Total packets: 1303
   Packet Size is : 1610 bytes
    Data rate: 215116.37 bytes/second
    Packet rate: 133.66 packets/second
  Flow 7:
    Total bytes: 2097152
    Total packets: 1865
   Packet Size is : 1125 bytes
    Data rate: 142611.73 bytes/second
    Packet rate: 126.82 packets/second
  Flow 8:
    Total bytes: 2097152
    Total packets: 1440
   Packet Size is : 1457 bytes
    Data rate: 280330.84 bytes/second
    Packet rate: 192.49 packets/second
  Flow 9:
    Total bytes: 2097152
    Total packets: 1880
   Packet Size is : 1116 bytes
    Data rate: 233036.19 bytes/second
    Packet rate: 208.91 packets/second
  Flow 10:
    Total bytes: 2097152
    Total packets: 1916
   Packet Size is : 1095 bytes
    Data rate: 188189.19 bytes/second
    Packet rate: 171.93 packets/second
  Overall statistics:
    Average data rate: 207321.69 bytes/second
    Average packet rate: 148.14 packets/second
abode@abode-virtual-machine:~/Downloads/pythonProject$
```





מסקנה:

כאשר עשינו הרצות עם מספרי זרימות 3,5,10 ראינו שקצב הנתונים הכולל יורד ככל שמגדילים את מספר הזרימות כי ככל שמגדילים את מספר הזרימות גודל ה data הכולל גדל ביחד עם מספר החבילות ולכן העומס גדל (כל הזרימות גודלן MB2).ניתן לראות זה בתוצאות שלנו כאשר ממוצע העברת המידע היה (Total Average Data Rate):

3 זרימות = 299958 (bytes/sec).

5 זרימות = 234029 (bytes/sec).

10 זרימות = 207321 (bytes/sec).

ושמנו לב שאותו הדבר קורה עבור ממוצע העברת החבילות הכולל (Total Average Packet Rate):

3 זרימות = 196.79 (packet/sec).

5 זרימות = 159.93 (packet/sec).

10 זרימות = 148.14 (packet/sec).

*ניתן לראות ששני הממוצעים יורדים ככל שגדל מספר הזרימות.

יש להוסיף שהנתונים שלנו מתבססים על המימוש כאשר עשינו שכל זרימה תהיה MB2 ובכל זרימה מוגרל באופן אקראי גודל כל חבילה משתנה (1000-2000 ביטים) הדבר שמשנה את כמות החבילות בכל זרימה בודדת ובכל הרצה עם מספר זרימות שונה.

לסיכום אנו חושבים שהפתרון שלנו טוב אך לא הכי יעיל אבל משקף את היחס של מספרי הזרימות לבין ממוצעי ה data/חבילות הכולל. הפרויקט היה מאתגר כאשר נתקלנו בהתחלה בבעיית אובדן חבילות אך לבסוף הצלחנו. זה מימוש פשוט של פרוטוקול quic ולא מכיל את כל תכונותיו כמו טיפול באיבוד מידע, בקרת העומס ,בטיחות לא נדרשנו).

חלק יבש

1 תארו במילים שלכם 5 חסרונות/מגבלות של TCP

בקרת עומס (CC) בTCP-מגבלת זרימת החבילות כדי למנוע עומס. CCמגביל את כמות החבילות שיכולה להישלח על ידי השולח כדי למנוע עומס ברשת. כאשר חלון העומס מצטמצם בעקבות אובדן חבילות, המהירות שבה ניתן לשלוח מידע יורדת. הדבר גורם לעיכובים בקבלת המידע בצד המקבל. לדוגמה, אם חלון העומס מצטמצם לגודל של 8 חבילות לאחר אובדן, המערכת תשלח פחות מידע בכל מחזור, מה שמגדיל את זמן הגעת המידע.

HEAD OF LINE BLOCKING: כאשר חבילת מידע אחת נאבדת בתור החבילות, כל התור נעצר עד לקבלת המידע הנאבד. הדבר יכול להוביל לעיכובים משמעותיים במעבר הנתונים, במיוחד בתקשורת בזמן אמת או בתקשורת מולטימדיה. לדוגמה, אובדן חבילת מידע אחת בתור יכול לעכב את קבלת כל החבילות האחרות בתור עד שהחבילת המידע הנאבדת תשלח מחדש ותתקבל בהצלחה.

עיכובים בהגדרת חיבור. ב TCP-ישנה דרישה להקים חיבור דרך תהליך של שלוש חבילות .(three-way handshake) תהליך זה כולל שליחת חבילת SYN, קבלת חבילת SYN-ACKושליחת חבילת .ACK תהליך זה יכול לגרום לעיכובים נוספים, במיוחד כאשר נעשה שימוש בפרוטוקולים כמו RT-TLS המחייבים שליחת חבילות נוספות לאישור החיבור המאובטח.

מגבלותTCP HEADER השדות seq# ו ACK#-ב TCP HEADER-מוגבלים ל-32 סיביות. המשמעות היא שהרצף המקסימלי של החבילות יכול להיות עד 4,294,967,295. הגבלה זו מגבילה את גודל חלון החבילות והביצועים של התקשורת, במיוחד ברשתות בעלות מהירויות גבוהות. הגדלת גודל השדות הללו יכולה לשפר את הביצועים ולהפחית את הסיכון לאובדן חבילות.

שימוש ב IP ברשת NAT יכול לסבך את ניהול החיבורים. כאשר מספר משתמשים חולקים כתובת IP ציבורית אחת, יש צורך בתרגום כתובות (NAT) עבור כל חבילת מידע. תהליך זה דורש תחזוקה של טבלת NAT אשר מתעדת את כתובות ה IP-הפנימיות המותאמות להן. הדבר עלול להוביל לעיכובים נוספים ולבעיות בקישוריות, במיוחד כאשר יש כמות גדולה של חיבורים פעילים בו זמנית.

2 ציינו 5 תפקידים שפרוטוקול תעבורה צריך למלא

לפי המאמר אפשר להבין (ב-2.5§) שחמישה התפקידים שפרוטוקול תעבורה צריך למלא הם:

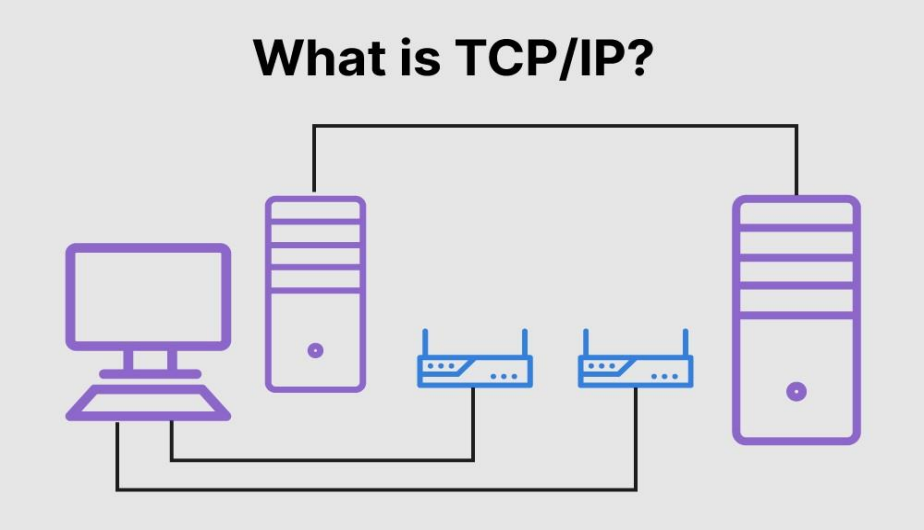
1 - הגדרת מזהי חיבור ומזהי נתונים: מזהים ייחודיים למעקב וניהול החיבורים.

2 - ניהול חיבורי תעבורה: תהליכים להקמה, תחזוקה וסיום של חיבורים ברשת, כולל התאמה לשינויים בכתובות.

3 - מסירת נתונים אמינה: הבטחת קבלה נכונה ומסודרת של נתונים באמצעות מנגנוני בקרה זרימה.

4 - בקרת עומס: טכניקה לשלוט על מספר החבילות ברשת, למניעת עומס ברשת.

5 - אבטחה: הגנה על הנתונים באמצעות הצפנה ואימות, להבטחת סודיות ושלמות.



3 תארו את אופן פתיחת הקשר (לחיצת ידיים") ב-QUIC. כיצד הוא משפר חלק מהחסרונות של TCP שתיארתם בסעיף 1.

אופן פתיחת הקשר:

- אם כבר נוצר חיבור קודם

הלקוח יכול לשלוח נתונים מוצפנים מהחבילה הראשונה, באמצעות ההגדרות מהחיבור הקודם. זה מאפשר לשלוח נתונים באופן מיידי, מבלי להמתין עד להשלמת לחיצת היד .

- אם זו הפעם הראשונה

הלקוח שולח חבילה ראשונית המכילה את מידע התעבורה. הסרבר מגיב עם חבילה ראשונית המכילה מידע תעבורה משלו והודעות אבטחת שכבת תעבורה. מידע הצפנה ותעבורה מוחלפים במקביל, מה שמאפשר ליצור את החיבור המאובטח בו זמנית .

TCP: 2 RTT בשידור הראשון לפחות
QUIC : 1 RTT בשידור הראשון או אם כבר היה חיבור אז 0

שיפור מחלק מהחסרונות של TCP:

QUIC מפחית את לחיצת היד המאובטחת ל RTT-יחיד על ידי שילוב הצפנה ואימות בפרוטוקול התעבורה, בניגוד ל TLS 1.3-על TCP

QUIC משתמש באישורי תעבורה כדי לשמור על קשרים למרות שינויים בכתובת IP, מה שמקל על ניידות. בניגוד ל TCP, שהוא תלוי כתובת IP ויציאה QUIC, מאפשר חיבורים רציפים במהלך שינויים ברשת.

QUIC תומך ב-0 RTT , המאפשר שליחת נתונים מתחילת לחיצת היד, ומפחית את זמן החיבור. בניגוד ל-TCP שדורש החלפה ראשונית כדי ליצור חיבור לפני שניתן לשלוח נתונים, בדרך כלל מעורבים 1-2 RTTs.

אם חבילה ראשונית אבדה , QUIC מאפשר ללקוח לשלוח מחדש חבילת שחזור עם טוקן רנדומאלי כדי לאמת את בקשת החיבור, ובכך לטפל באיבוד חבילות בצורה יעילה יותר מאשר TCP שגורמת לעיכובים משמעותיים.

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

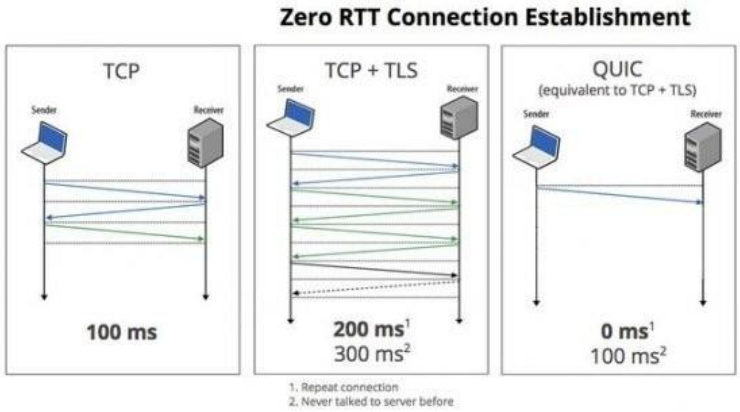
בניגוד ל TCP שיש לה פורמט Headers קבוע, Quic משתמש בשני סוגים כותרות של פקטות:

Headers ארוכות: בשימוש ברגע של יצירת חיבור, הן מכילות מידע הדרוש להפעלת החיבור

Headers קצרות: בשימוש ברגע שהחיבור נוצר, הן מכילות רק את השדות הדרושים לתקשורת יעילה .

לכל חבילת Quic יש מספר ייחודי המציין את סדר השידור, מה שמקל על התאוששות מהפסדים ולדעת במדויק כמה פקטות יש ברשת. הפורמט הגמיש והניתן להתאמה של Quicעם כותרות ספציפיות למצבי חיבור שונים ומספרי פקטות ייחודיים, משפר את ניהול הפקטות ואת עמידות האובדן בהשוואה לפרוטוקול בקרת שידור.

QUIC שולח ACKs הכוללים מספרי רצף מרובים של חבילות שהתקבלו, מה שמפחית את מספר החבילות הדרושות לאישור קבלתם.
ACK מאפשר עד 256 בלוקים של ACK בהודעה אחת, משפר את הניהול של חבילות אבודות בהשוואה ל-TCP.
QUIC מנהל מספר זרמי נתונים בו-זמנית באותו חיבור, כמו HTTP/2.
זרימות בלתי תלויות QUIC נמנעות מחסימת head of the line, ומאפשרות לזרימות אחרות להמשיך כרגיל גם אם לזרימה אחת יש בעיות.



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

כאשר חבילה ב QUIC-מגיעה באיחור או לא מגיעה כלל, הפרוטוקול משתמש במספר מנגנונים כדי לטפל במצב זה ולמזער את ההשפעה על הביצועים והאמינות של התקשורת.

ראשית, כל חבילה ב QUIC-מכילה בתוכה מספר מסגרות .(frames) כאשר חבילה מקבלת אישור קבלה ,(ACK) כל המסגרות שבתוכה נחשבות כאילו שהגיעו בהצלחה. אם חבילה לא מקבלת ,ACK והחבילות שנשלחו אחריה כבר קיבלו ,ACK המסגרות שבתוכה נחשבות כאבודות.

QUICמשתמש בשני סוגים של סף כדי לקבוע מתי חבילה נחשבת כאבודה. הסף הראשון מבוסס על מספר החבילות: אם המספר הסידורי של חבילה נמוך בכמות מסוימת מזו שקיבלה ,ACK היא נחשבת כאבודה. הסף השני מבוסס על זמן: חבילה שנשלחה לפני זמן רב יותר מה RTT-המשוער ולפני חבילה שקיבלה ,ACK נחשבת כאבודה.

כאשר חבילה נחשבת כאבודה QUIC ,מתמודד עם המצב על ידי שליחה מחדש של המסגרות האבודות בחבילה חדשה. בנוסף, הפרוטוקול מתמודד גם עם חבילות משוכפלות, מה שמפחית את כמות החבילות הנשלחות מחדש שלא לצורך QUIC .כולל גם מנגנונים המאפשרים לחבילה שנשלחת עם ACK לכלול את ה RTT-המשוער והשונות, מה שמונע שליחה מחדש של חבילות מיותרות.

בנוסף לכך QUIC ,משתמש בטיימרים שמסייעים לקבוע מתי יש לשלוח חבילה מחדש. כאשר הזמן המקסימלי המותר לקבלת ACK חולף ללא קבלת ,ACK השולח ישלח חבילה חדשה לבדיקה. פעולה זו מאפשרת להפחית את כמות החבילות הנשלחות מחדש ומסייעת לשמור על הסדר והאמינות של התקשורת.

במסקנה, QUIC מטפל באובדן חבילות ובאיחורים בצורה יעילה על ידי שימוש במנגנונים לזיהוי אובדן, שליחה מחדש חכמה של חבילות, והתמודדות עם חבילות משוכפלות. כך, QUIC מצליח לספק סדר ואמינות בתקשורת, בדומה ל-TCP, אך עם יעילות משופרת.

6 תארו את בקרת העומס (congestion control) של QUIC.

QUIC, משתמש במספר פקטורים לבקרת עומסים כמו ה offset-וה frames-ב-streamsלצורך בקרה אמינותית. בעת עומס QUIC, מתאים את קצב השליחה כדי למנוע יתר עומס ברשת, על ידי מתן אפשרות לחבילות אחרות להמשיך לזרום ללא הפרעה, גם אם חבילה אחת מתעכבת .

בזמן עומס QUIC, נותנת לשולח חופש לפעול על פי מדדי העומס הקיימים. השולח בודק את ההכרחי של חלון הזרימה ולא מצמצם אותו באופן אוטומטי. הוא מוודא שאין שום חבילה שעמוסה ובמקרה של זרימה עמוסה, ממתין לפני צמצום חלון הזרימה .

במקרה של חבילות שאבדו או מתעכבות QUIC, שולחת מחדש את החבילות בצורה יעילה על ידי שמירת העומס הקצר דרך שליחת הפקטות שמתאימות על בסיס הזמן של ה RTT-המחושב וגודל הפקטה.

השילוב הזה בין ניהול דינמי של הזרימה לבין הבקרה על העומס ברשת מאפשר ל-QUIC להציע יתרונות משמעותיים בהשוואה ל-TCP, כולל שמירה על ביצועים גבוהים ואמינות גבוהה גם בתנאי רשת עמוסים.